Securing information systems against advanced persistent threats (APTs).

EKE, E.N.

2024

Securing Information Systems against Advanced Persistent Threats (APTs)

Hope Nkiruka Eke

PhD 2024

Securing Information Systems against Advanced Persistent Threats (APTs)

Hope Nkiruka Eke

A thesis submitted in partial fulfillment of the requirements of

Robert Gordon University

for the degree of Doctor of Philosophy

January 2024

Declaration of Authorship

I, Hope Nkiruka Eke, declare that this thesis titled, "Securing Information Systems against Advanced Persistent Threats (APTs)" submitted by me, for the award of the degree of *Doctor of Philosophy* to Robert Gordon University is a record of bonafide work carried out independently by me under the supervision of Dr. Andrei Petrovski, School's Research Degree Coordinator, & Dr. Hatem Ahriz (School of Computing, Robert Gordon University Aberdeen, United Kingdom), and Dr M. Omar Al-Kadri (Birmingham City University, Birmingham, B4 7XG). This thesis has not been submitted for the award of any other degree or professional qualification.

I further declare that part of the work reported in this thesis has appeared in the following publications:

- Eke, Hope Nkiruka., & Petrovski, Andrei. (2023, May). Advanced Persistent Threats Detection based on Deep Learning Approach. In 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS) (pp. 1-10). IEEE. Available from: https://doi.org/10.1109/ICPS58381.2023.10128062
- Eke, Hope Nkiruka, Andrei Petrovski, Hatem Ahriz, and M. Omar Al-Kadri. (2022). "Framework for Detecting APTs Based on Steps Analysis and Correlation." In Security and Resilience in Cyber-Physical Systems, pp. 119-147. Springer, Cham, 2022. Available from: https://doi.org/10.1007/978-3-030-97166-3_6
- Eke, Hope, Andrei Petrovski, and Hatem Ahriz (2020a). "Detection of false command and response injection attacks for cyber physical systems security and resilience". In: 13th International Conference on Security of Information and Networks, pp 1-8. Available from: https://doi.org/10.1145/3433174.3433615
- Eke, Hope, Andrei Petrovski, Hatem Ahriz (2020b). "Handling minority class problem in threats detection based on heterogeneous ensemble learning approach". In: International Journal of Systems and Software Security and Protection (IJSSSP) 11(2), pp. 13-37. Available from: https://doi.org/10.4018/IJSSSP.2020070102
- Eke, Hope Nkiruka, Andrei Petrovski, and Hatem Ahriz (2019). "The use of machine learning algorithms for detecting advanced persistent threats". In: Proceedings of the 12th International Conference on Security of Information and Networks, pp. 1-8. Available from: https://doi.org/10.1145/3357613.3357618

Place: Aberdeen, UK

Date: January 16, 2024,

Hope Nkiruka Eke

Repository note: this page intentionally left blank.

Certificate

This is to certify that this thesis titled "Securing Information Systems against Advanced Persistent Threats (APTs)" submitted by Ms. Hope Nkiruka EKE, School of Computing, RGU Aberdeen Scotland, United Kingdom for the award of the degree of *Doctor of Philosophy*, is a record of bonafide work carried out by her under my supervision, as per the RGU code of academic and research ethics.

Part of the contents of this report has appeared in the following publications:

- Eke, Hope Nkiruka., & Petrovski, Andrei. (2023, May). Advanced Persistent Threats Detection based on Deep Learning Approach. In 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS) (pp. 1-10). IEEE. Available from: https://doi.org/10.1109/ICPS58381.2023.10128062
- Eke, Hope Nkiruka, Andrei Petrovski, Hatem Ahriz, and M. Omar Al-Kadri. (2022). "Framework for Detecting APTs Based on Steps Analysis and Correlation." In Security and Resilience in Cyber-Physical Systems, pp. 119-147. Springer, Cham, 2022. Available from: https://doi.org/10.1007/978-3-030-97166-3_6
- Eke, Hope, Andrei Petrovski, and Hatem Ahriz (2020a). "Detection of false command and response injection attacks for cyber physical systems security and resilience". In: 13th International Conference on Security of Information and Networks, pp 1-8. Available from: https://doi.org/10.1145/3433174.3433615
- Eke, Hope, Andrei Petrovski, Hatem Ahriz (2020b). "Handling minority class problem in threats detection based on heterogeneous ensemble learning approach". In: International Journal of Systems and Software Security and Protection (IJSSSP) 11(2), pp. 13-37. Available from: https://doi.org/10.4018/IJSSSP.2020070102
- Eke, Hope Nkiruka, Andrei Petrovski, and Hatem Ahriz (2019). "The use of machine learning algorithms for detecting advanced persistent threats". In: Proceedings of the 12th International Conference on Security of Information and Networks, pp. 1-8. Available from: https://doi.org/10.1145/3357613.3357618

This thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place: Aberdeen, UK

Date:

(Dr. Andrei Petrovski)

Repository note: this page intentionally left blank.

Abstract

Advanced Persistent Threats (APTs) have been a major challenge in securing both Information Technology (IT) and Operational Technology (OT) systems. APTs are sophisticated attacks that masquerade their actions to navigate around defenses, breach networks, often, over multiple network hosts and evades detection. It also uses "low-and-slow" approach over a long period of time. While APTs have drawn increasing attention from the industrial security community, recent security products are inadequate at helping companies defend against APTs attacks due to APTs' prolonged, stealthy characteristics and sophisticated levels of expertise and significant resources. The current APTs best practice requires a wide range of security countermeasures resulting in a multi-step detection approach which opens new research directions. The detection of a single step of APT lifecycle does not infer detection of a complete APT full scenario. The accurate detection and prevention of APT in real time is an ongoing challenge.

This research aims to investigate APT attack detection and develop a novel multi-step APT attack detection framework to detect APT attack steps. An APT steps analysis and correlation framework termed ***"APT$_{DASAC}$"*** is proposed. This approach takes into consideration the distributed and multi-level nature of industrial control system (ICS) architecture and reflects on multi-step APT attack lifecycle. The implementation is carried out in three stages: stage one is "Data input and probing layer", this involves data gathering and pro- cessing; the second stage is "Data analysis & Correlation layer", this stage applies the core process of APT_{DASAC} to learn the behaviour of attack steps from the sequence data, corre- late and link the related output; and stage three "Decision layer", the ensemble probability approach is utilized to integrate the output and make attack prediction. The framework was validated with four different datasets and four case studies: i) network transactions between a remote terminal unit (RTU) and a master control Unit (MTU) in-house super- visory control and data acquisition (SCADA) gas pipeline control system, ii) a case study of command and response injection attack, iii) a scenario based on network traffic containing hybrid of the real modern normal and the contemporary synthesized attack activities of the network traffic and iv) APT_alerts - a historic record of APT alert generated through a monitored network. The system achieved the probability average prediction accuracy of **86.73%**. It also achieved a significant *detection_rate* of **93.50%, 80.98%, 85.19%, & 80.90%** for each individual APT lifestyle detectable steps (A, B, C,& D). Experimentally, $APT_{D\ AS\ AC}$ achieved a significant attacks detection capability and demonstrated that attack detection techniques applied that performed very well in one domain may not yield the same good result in another domain. This suggests that robustness and resilience of operational systems state to withstand attack and maintain system performance and resilience are de- termined by the safety and security measures in place, which is specific to that system in question.

Keywords: *Advanced Persistent Threats*, *Cyber Attacks*, MITRE ATT&CK, *Intrusion, Detection, Techniques.*

Repository note: this page intentionally left blank.

Acknowledgements

With immense pleasure and deep sense of gratitude, I wish to express my sincere thanks to my supervisors **Dr. Andrei Petrovski**, School's Research Degree Coordinator, **Dr. Hatem Ahriz**, Senior Lecturer, School of Computing, Robert Gordon University, and **Dr. M. Omar Al-Kadri**, external supervisor, Birmingham City University, Birmingham, B4 7XG, without their motivation, support and continuous guide, this research would not have been successfully completed.

I am grateful to the Chancellor of Robert Gordon University (RGU), **Professor Paul Hagan**, the Vice Chancellor **Professor Steve Olivier** and RGU for providing infrastructural facilities and many other resources needed for my research. I express my sincere thanks to **staff** of School of Computing for their help and **Dr. John Isaacs**, Head of School of Computting, RGU for his kind words of support and encouragement. I like to acknowledge the support and feedback rendered by **my colleagues** in several ways throughout my research work.

My heart felt gratitude goes to **Ms. Ogochukwu Ikenwilo** for all her support and encouragement. **Dr. Peter Okay** & family, **Dr. Bartholomew Aleke** and **Mrs Roseline Johnson** for their moral support. I appreciate the support and prayers of **(Brother David** & **Kudzi Danfag**, **Sister Kam**, **Brother Andy** & **Uche Makoni**, **Sister Rachael** & **Isaac Foo** and family of **Kings Community Church Aberdeen)**. To all my friends and families too numerous to mention, I appreciate all your support.

I wish to extend my profound gratitude to my mother **Mrs Comfort Onwe** for all the sacrifices she made during my research and also providing me with moral support and encouragement whenever required.

Last but not the least, special thanks to my daughter **Zoe Amarachi Adanna Eke** and my sons **Donald Emeka Eke** and **Joel Ebuka Eke** for their constant encouragement and moral support along with patience and understanding especially when I needed time for my study.

Place: Aberdeen, UK

Date: January 16, 2024

Hope Nkiruka Eke

Repository note: this page intentionally left blank.

Contents

***

Declaration of Authorship iii

Certificate v

Abstract vii

Acknowledgements ix

List of Figures . . . xvi
List of Tables . . . xviii
List of Algorithm . . . xix
Listings . . . xxi
List of Abbreviations . . . xxiii
List of Symbols . . . xxvii
List of Symbols . . . xxviii

1 Introduction 1
1.1 Overview of Cyber Security in Current Era . . . 1
1.2 Research Motivation . . . 3
1.3 Problem Statement . . . 4
1.4 Research Aims . . . 4
1.4.1 Research Questions . . . 4
1.4.2 Research Objectives . . . 5
1.5 Contributions . . . 6
1.5.1 Research Outputs Links . . . 8
1.5.1.1 List of Publications . . . 8
1.6 Thesis Outline . . . 9
1.7 Conclusion . . . 9

2 Background and Related Work 11
2.1 Advanced Persistent Threats (APTs) . . . 11
2.1.1 Defining APT . . . 11
2.1.2 APTs Traits . . . 12
2.1.3 An Overview of APT Lifecycle . . . 13
2.1.4 APT Intent . . . 15
2.1.5 Examples of known APT Attacks . . . 16
2.2 Cybersecurity Countermeasures & Mitigation . . . 17
2.3 Industrial Control System (ICS) . . . 18
2.3.1 ICS Security Objectives . . . 19
2.3.2 Security Issues . . . 20
2.3.3 Stealth Cyber Attacks . . . 21

2.3.4 ICS Cyber Security Design Tools . . . 24
2.3.4.1 The Purdue Model for ICS Architecture . . . 25
2.3.4.2 MITRE ATT&CK Framework . . . 26
2.4 Intrusion Detection and Prevention Systems (IDPSs) . . . 28
2.4.1 Why Deploying IDPSs Technology? . . . 29
2.4.2 IDPSs Common Detection Methodologies . . . 30
2.4.2.1 Anomaly-Based Methodology . . . 30
2.4.2.2 Signature-Based Methodology . . . 31
2.4.2.3 Stateful Protocol Analysis Based Methodology . . . 32
2.4.2.4 Hybrid-Based Methodology . . . 34
2.4.3 IDPSs Common Technologies . . . 34
2.4.3.1 Network-based IDPS (N-IDPS) Technology . . . 35
2.4.3.2 Host-based IDPS (H-IDPS) Technology . . . 35
2.4.3.3 Wireless-based IDPS (W-IDPS) Technology . . . 36
2.4.3.4 Network Behaviour Analysis (NBA) IDPS Technology . . . 36
2.4.3.5 Hybrid/Distributed-Based IDPS Technology (HD-IDPS) Technology . . . 37
2.5 Deep Neural Network (DNN) . . . 37
2.5.1 Recurrent Neural Network (RNN) . . . 39
2.5.2 Long Short-Term Memory (LSTM) . . . 42
2.5.3 Gated Recurrent Unit (GRU) . . . 44
2.5.4 Decision Tree Classifier (DTC) . . . 44

3 Detection and Defence of APTs Attack Approaches 47
3.1 APT Detection Methods . . . 47
3.1.1 The ATT&CK Matrix Techniques Tactics Procedures (TTPs) Concept 48
3.1.2 Useful Outgoing Network Traffic Attributes to Uncover Malicious Activities . . . 48
3.2 Existing Proposed APT Detection Systems . . . 49
3.2.1 APT Attack Detection and Prevention . . . 50
3.2.2 Defending APT Attack . . . 53
3.2.3 APT Attack Mitigation . . . 54
3.2.4 Deep Learning Detection Approach . . . 55
3.3 Existing APT Detector's Limitations . . . 56
3.3.1 Summary of Reviewed Proposed APT Detection Systems . . . 58
3.4 Summary of the Reviewed Detection Methods . . . 59
3.5 Experimental Datasets . . . 61
3.5.1 APT_alerts_dataset.db . . . 61
3.5.2 The New Gas Pipeline (NGP) Dataset . . . 62
3.5.2.1 Three Main Features of NGP Dataset . . . 62
3.5.2.2 Cyber Attacks as contained in the NGP data Records . . . 62
3.5.2.3 Identified Cyber Threats in NGP Dataset . . . 64
3.5.2.4 Raw Dataset . . . 64
3.5.3 UNSW-NB15 Dataset . . . 65
3.5.3.1 UNSW-NB15 Data Attack Type and Description . . . 66
3.5.4 KDDCup'99 Dataset . . . 67
3.5.4.1 KDDCup'99 and NLS-KDD Datasets Limitations . . . 67
3.5.4.2 KDDCup'99 Dataset Statistical Analysis Representation . . . 68
3.5.5 Comparison of Experimental Datasets . . . 69
3.6 Preliminary Experiments . . . 69
3.6.1 Application of Machine Learning Algorithms for Threats Detection . 69

3.6.1.1 Experiment One . . . 71
3.6.1.2 Results and Discussions - Experiment One . . . 71
3.6.2 Handling Minority Problem in Threats Detection . . . 74
3.6.2.1 Experiment Two . . . 75
3.6.2.2 Results and Discussions - Experiment Two . . . 76

4 APT Detection Framework Based on APT Steps Analysis and Correlation 81
4.1 Rational of Design . . . 81
4.2 APT_{DASAC} Architectural Design . . . 82
4.3 The Three Stages of APT_{DASAC} . . . 83
4.3.1 Data Input and Probing Layer . . . 83
4.3.2 Analysis & Correlation Layer . . . 86
4.3.3 APT_{DASAC} Attack Defence Points . . . 89
4.3.3.1 Step 1 - Reconnaissance and Weaponization (R&W) . . . 89
4.3.3.2 Step 2 - Delivery (D) . . . 89
4.3.3.3 Step 3 - Initial Intrusion and Exploitation (II&E) . . . 90
4.3.3.4 Step 4 - Lateral Movement and Operation (LM&O) . . . 90
4.3.3.5 Step 5 - Data Discovery and Collection (DC) . . . 91
4.3.3.6 Step 6 - Data Exfiltration (DE) . . . 92
4.3.4 Decision Layer . . . 92
4.4 Summary . . . 93

5 Implementation of APT_{DASAC} 95
5.1 Implementation Setup . . . 95
5.2 Hyperparameters Settings Used . . . 96
5.3 Implementation of the Detection Framework . . . 97
5.4 Detection Module . . . 97
5.4.1 Preparing the Used Dataset . . . 97
5.4.2 Implementation of Alert Filtering, Clustering and Correlation Modules 101
5.4.2.1 Implementation of Alert Filtering Module (AFM) . . . 101
5.4.2.2 Implementation of Alert Clustering & Correlation Module (ACCM) 101
5.4.3 Training the Detection Model . . . 105
5.4.4 Applying the Ensemble Module for Detection . . . 105
5.5 Summary . . . 105

6 Evaluation Results 107
6.1 Evaluation Metrics . . . 107
6.2 Experimental Evaluation of APT_{DASAC} Setup . . . 109
6.3 Evaluation of Application Domains . . . 109
6.3.1 Case Study One Evaluation – Application to The New Gas Pipeline Control System Dataset . . . 111
6.3.2 Case Study Two Evaluation – Application to KDDCup'99 Dataset . . . 114
6.3.3 Case Study Three Evaluation – Application to UNSW-NB15 Dataset . . . 114
6.3.4 Case Study Four Evaluation – Application to APT_alerts_dataset.db . . . 117
6.3.5 Individual Detectable Steps Evaluation Result . . . 118
6.4 Performance Evaluation of APT_{DASAC} . . . 120
6.5 Results and Discussion . . . 121
6.5.1 Model Performance Comparison . . . 139
6.5.2 APT_{DASAC} Comparison to other Existing APT Detection Systems . . . 142
6.6 Mitigation of APTs at different Stages in the Lifecycle . . . 145
6.6.1 APTs Lifecycle Mitigation Approach . . . 145
6.7 Summary . . . 147

7 Conclusion **149**
7.1 Summary . . . 149
7.1.1 Contributions Summary . . . 150
7.2 Limitations and Future Work . . . 152
7.2.1 Limitations . . . 153
7.2.2 Future Work . . . 153
7.3 Conclusion . . . 154

Bibliography **157**

A Author's Publications **175**
A.1 Book Chapter . . . 175
A.2 Conferences . . . 175
A.3 Journals . . . 175
A.4 Symposium . . . 175
A.5 Poster . . . 176

B Derivative of Los Function **177**

C Codes and Snippets **179**
C.1 Code Snippets . . . 179
Index . . . 187

List of Figures

***

1.1 An example of a simple SCADA system 2

2.1 Advanced Persistent Threat (APT) 13
2.2 APT Lifecycle 15
2.3 Security Objectives related to an ICS 19
2.4 Classification of cyber stealth attack types and sub-types 21
2.5 Five Techniques to Achieve an Attacker's Goal 27
2.6 Pre-ATT&CK and Enterprise ATT&CK 28
2.7 The Architecture of an IDPS 29
2.8 Architecture of an Anomaly-Based Methodology 31
2.9 The Architecture of Signature-Based Methodology 32
2.10 The Architecture of Stateful Protocol Analysis-Based Methodology 33
2.11 The Architecture of a Hybrid-Based Methodology 34
2.12 Schema of Unfolded Basic Recurrent Neural Network [51] 40

3.1 SPuNge Architecture 51
3.2 Multi-Stage Attack Framework Overview 52
3.3 MalNet Architecture 56
3.4 APT_alerts_dataset Records Distribution 61
3.5 APT_alerts Log Sample with Attributes 62
3.6 NGP Data Records Distribution 63
3.7 Four Main Attack Group and Normal Classes 63
3.8 The instances within NGP raw dataset 64
3.9 UNSW-NB15 train dataset 65
3.10 UNSW-NB15 test dataset 66
3.11 10% KDDCup'99 Data Records Distribution 68
3.12 Binary & Multi-Classification Confusion Matrix for all the Implemented Algorithm [51]. 73
3.13 10% KDDCu99 dataset confusion matrix of 5 classes with and without SMOTE oversampling techniques applied [1]. 77
3.14 Overall summary performance result of class classification of both resampled and non-resampled data for 10% KDDCup99 and UNSW-NB15 datasets [1] 78
3.15 The visual representation of each algorithm's validation accuracy and loss rate on each iteration on 10% KDDCup99 dataset with SMOTE oversampling techniques applied [1]. 79
3.16 The visual representation of each algorithm's validation accuracy and loss rate on each iteration on 10% KDDCup99 dataset without resampling techniques applied [1]. 79

4.1 APT detection framework based on APT steps analysis & correlation (APT_{DASAC}) 82

5.1 Transformed Data 98

6.1 t-SNE 3D Visualization Projection of NGP, UNSW-NB15 and KDDCup'99 Datasets Records Distribution 110
6.2 Classification report for all the algorithms on NGP_5 dataset 112
6.3 Classification report for all the algorithms on NGP_6 dataset 113
6.4 Classification report for all the algorithms on KDDCup'99 dataset 115
6.5 Classification report for all the algorithms on UNSW-NB15 dataset 116
6.6 Result report for APT_{DASAC} & ML-DT on APT_alert_dataset Records 117
6.7 Four Detectable APT_ALert Steps 118
6.8 Individual Detectable Steps Detection Result 119
6.9 Four Main Attack Types and Normal Records for NGP Data 123
6.10 Validation Accuracy and Loss rate against Epochs for NGP Dataset 124
6.11 The visual Representation of AUC-ROC graph for all the main four attack types and normal records for of all the classes and minority ***"class 3"*** for NGP dataset 125
6.12 Confusion Matrix of NGP_6 Dataset Records 126
6.13 Validation Accuracy and Loss rate against Epochs for NGP_6 Dataset 127
6.14 The visual representation of AUC-ROC graph for all the attack types and normal records and minority ***"class 3"*** for NGP_6 dataset 128
6.15 Confusion Matrix of KDDCup'99 Dataset Records 131
6.16 Validation Accuracy and Loss rate against Epochs for KDDCup'99 Data . . . 132
6.17 The visual Representation of AUC-ROC graph for all classes and ***"class 0"*** for KDDCup'99 Data 133
6.18 Confusion Matrix of UNSW-NB15 dataset Records 135
6.19 Validation Accuracy and Loss rate against Epochs for UNSW-NB15 Data . . 136
6.20 The visual Representation of AUC-ROC graph for all classes and ***"class 4"*** for UNSW-NB15 Dataset 137
6.21 Confusion Matrix for APT_alert_dataset Records 138
6.22 The visual representation of AUC-ROC graph for all the detectable attack steps for APT_alert_dataset 138
6.23 Algorithm Comparison - NGP_6 Data 139
6.24 APTDASAC and ML-DT Comparison - NGP_6 Data 140
6.25 Algorithm Comparison - KDDCup'99 Data 141
6.26 APT_{DASAC} and ML-DT Comparison - KDDCup'99 Data 141
6.27 Algorithm Comparison - UNSW-NB15 Data 141
6.28 APT_{DASAC} and ML-DT Comparison - UNSW-NB15 Data 142
6.29 APT_{DASAC} and ML-DT detection capability on the four detectable steps of APT_alert_dataset.db 144

List of Tables

***

2.1 Two Significant ways APT differ from Traditional Threat 15
2.2 Differences among Basic, Traditional Attack and APT 16
2.3 Purdue Model for ICS Infrastructure 25

3.1 Summary of Reviewed APT Detectors 58
3.2 Attack Categories with Normal Records Type 63
3.3 UNSW-NB15 Data Attack Type and Description 66
3.4 KDDCup'99 Dataset Attack and Normal Instances Distribution 68
3.5 Comparison of the four Experimental Datasets 69
3.6 Expanded Algorithms Names - the list of expanded names of all the algorithms as used for this experiment are shown on this table [51]. 71
3.7 Average Binary Summary Results - comparative summary result of the ML algorithms, the LSTM-RNNs network and the result of previously published AIS (NS-AFB to be more precise) [51]. 72
3.8 Summary Results of Average Multi-Class - this table contains the output summary of the performance of each of the algorithms in detecting the four main attack groups [51]. 72

4.1 APT_{DASAC} Detectable & Non-Detectable Steps of APT Lifecycle 88
4.2 APT lifecycle detectable attack steps / alert as contained in APT_alerts_dataset.db, and also discussed in Sub-section [2.1.3]. 89

6.1 The Confusion Matrix Description for Multiple Classes: with letter A–E representing each class of the detectable classes as contained within the APT_alerts_dataset.db, NGP_5 and KDDcup'99 datasets used. Additional columns and rows are required to generate confusion matrix for NGP_6 and UNSW-NB15 dataset (not shown on this table). 108
6.2 Summary of each individual detectable steps detection result of implementing APT_{DASAC} on APT_alerts_dataset.db evaluation. The mean average *Precision, Recall, f1-score, AUC_ROC curve* and overall *Detection_rate* of each individual detectable steps recorded against each step. The metrics parameter with best result for each individual step result are written in bold text. . 119
6.3 Summary of the Average Result Report of all Algorithms on NGP_5, NGP_6, UNSW-NB15 and KDDCup'99 Datasets Evaluated. With mean average Precision, Recall, f1-score, and overall average accuracy recorded for different problem domains networks cross evaluation. The metrics parameter with best result for each individual algorithm are written in bold text. 120

6.4 Overall Summary Result of all Algorithms on NGP_5, NGP_6, UNSW-NB15 and KDDCup'99 Datasets. With TPR, FPR, f1-score, and micro/macro-ave_roc curve recorded for different problem domains networks cross evaluation. The metrics parameter with best result for each individual algorithm are written in bold text. 121
6.5 The macro f1-score, error, f1-score, and Overall average model accuracy recorded for the different problem domains networks cross evaluation on these different sets. The first two metrics parameter with best result for each metrics against each dataset are written in bold text. 121
6.6 APT_{DASAC} Comparison to other Existing APT Detection Systems 144
6.7 *Measures for Mitigating APT Lifecycle* 146

List of Algorithms

***

1 Data Input and Probing Layer Pseudocode 86
2 Analysis Layer Pseudocode 87
3 Pseudocode for Delivery Point Attack Detection 91
4 Decision Layer Pseudocode 93

5 Data Pre-processing 99
6 Alert Filtering Module (AFM) 101
7 Detection Module Implementation - Alert Generation 103
8 Detection Module Implementation - Generated Alert Correlation 104

Repository note: this page intentionally left blank.

Listings

***

C.1 Used Libraries 179
C.2 Feature Transformation 180
C.3 Balancing Data Features 180
C.4 Data Normalization 180
C.5 Label Encoder 181
C.6 Confusion Matrix Function 181
C.7 Process_Data Module 182
C.8 Enumeration_Function Module 182
C.9 ROC for Single-Class Module 183
C.10 ROC for Multi-Class Module 183
C.11 Function to Plot Graphs for Loss and Accuracy 185
C.12 Prediction Graph Function for Multi-classes Module 185
C.13 Ensemble Module 186

Repository note: this page intentionally left blank.

List of Abbreviations

***

Acronyms	Description
ACCS	**A**ustralian **C**entre for **C**yber **S**ecurity
ACCM	**A**lert **C**lustering and **C**orrelation **M**odule
AFM	**A**lert **F**iltering **M**odule
AGC	**A**utomatic **G**eneration **C**ontrol **S**ystem
AI	**A**rtificial **I**ntelligent
AIS	**A**rtificial **I**mmune **S**ystem
ALBF	**A**ctive **L**earning **B**ase **F**ramework
ANN	**A**rtificial **N**eural **N**etwork
APT	**A**dvanced **P**ersistent **T**hreat
APTs	**A**dvanced **P**ersistent **T**hreats
APT_{DASAC}	**A**PT **F**ramework **B**ased on **A**PT **S**teps **A**nalysis and **C**orrelation
APT-AN	**A**PT - **A**ttack **N**etwork
APT-BN	**A**PT - **D**riven **B**ayes **N**et
ATT&CK	**A**dversarial **T**actics **T**echniques & **C**ommon **K**nowledge
AUC	**A**rea **U**nder the **C**urve
BiLSTM	**B**idirectional **L**ong **S**hort-**T**erm **M**emory
BOYD	**B**ring **Y**our **O**wn **D**evice
BPTT	**B**ack **P**ropagation **T**hrough **T**ime
CAD	**C**omputer **A**ided **D**iagnosis system
C&C	**C**ommand and **C**ontrol
CII	**C**ritical **I**nformation **I**nfrastructure
CI	**C**ritical **I**nfrastructure
CFS	**C**orrelation-based **F**eature **S**election
CEC	**C**onstant **e**rror **c**arousel
CMRI	**C**omplex **M**alicious **R**esponse **I**njection
DARPA	**D**efense **A**dvanced **R**esearch **P**rojects **A**gency
DCS	**D**istribution **C**ontrol **S**ystem

Acronyms	Description
DGA	**D**omain **G**eneration **A**lgorithm
DL	**D**eep **L**earning
DNN	**D**eep **N**eural **N**etworks
DNS	**D**omain **N**ame **S**ervice
DMZ	**D**emilitarized **Z**one
e.g.	**e**xample
etc.	**e**t **c**etera
FAR	**F**alse **A**larm **R**ate
FDI	**F**alse **D**ata **I**njection
FG	**F**orget **G**ates
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
FPR	**F**alse **P**ositive **R**ate
GCN	**G**raph **C**onvolutional **N**etworks
GDP	**G**ross **D**omestic **P**roduct
GNN	**G**raph **N**eural **N**etwork
GPU	**G**raphics **P**rocessing **U**nit
GRU	**G**ated **R**ecurrent **U**nit
HDFS	**H**adoop **D**istributed **F**ile **S**ystem
HD-IDPS	**H**ost/**D**istributed-**B**ased **I**DPS **S**ystem
H-IDPS	**H**ost-**B**ased **I**DPS **S**ystem
HMI	**H**uman **I**nterfaces
HMM	**H**idden **M**arkov **M**odel
HTTP	**H**ypertext **T**ransfer **P**rotocol
ICS	**I**ndustrial **C**ontrol **S**ystems
ICS-CERT	**I**ndustrial **C**ontrol **S**ystems **C**yber **E**mergency **R**esponse **T**eam
ID	**I**ntrusion **D**etection
IDS	**I**ntrusion **D**etection **S**ystem
IDPSs	**I**ntrusion **D**etection and **P**revention **S**ystems
i.e.	that is
IDS-NNM	**I**ntrusion **D**etection **S**ystems using **N**eural **N**etwork based **M**odelling
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IFT	**I**nformation **F**low **T**racking
IM	**I**nstant **M**essaging

Acronyms	Description
IP	**I**nternet **P**rotocol
IPs	**I**ntrusion **P**reventions
IPS	**I**ntrusion **P**revention **S**ystem
ISACA	**I**nformation **S**ystems **A**udit and **C**ontrol **A**ssociation
IT	**I**nformation **T**echnology
KDDCup'99	**K**nowledge **D**iscovery and **D**ata **M**ining **D**ataset
LANL	**L**os **A**lamos **N**ational **L**aboratory
LSTM	**L**ong **S**hort-**T**erm **M**emory
MAC	**M**edia **A**ccess**C**ontrol
MD5	**M**essage **D**igest Algorithm
MFCI	**M**alicious **F**unction **C**ode **I**njection
ML	**M**achine **L**earning
MLS-AC	**M**ulti **L**evel **S**ecurity-**A**ccess **C**ontrol
MPCI	**M**alicious **P**arameter **C**ommand **I**njection
MI	**M**yocardial **I**nfarction
MIME	**M**ultipurpose **I**nternet **M**ail **E**xtensions
MSCI	**M**alicious **S**tate **C**ommand **I**njection
MSAs	**M**ulti-stage **A**ttacks
MSSP	**M**anaged **S**ecurity **S**ervice **P**rovider
MTU	**M**aster **T**erminal **U**nit
MM-TBM	**M**arkov **M**ulti-**P**hase **T**ransferable **B**elief **M**odel
NBA	**N**etwork **B**ehaviour **A**nalysis
NE	**N**ash **E**quilibrium
NGP	**N**ew **G**as **P**ipeline **D**ataset
NIDPS	**N**etwork **I**ntrusion **D**etection and **P**revention **S**ystem
N-IDPS	**N**etwork-**B**ased **I**ntrusion **D**etection and **P**revention **S**ystem
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
NMRI	**N**aïve **M**alicious **R**esponse **I**njection
OT	**O**perational **T**echnology
OSINT	**O**pen **S**ource **I**ntelligence
P	**P**recision
PERA	**P**urdue **E**nterprise **R**eference **A**rchitecture
PID	**P**roportional **I**ntegral **D**erivative
PLA	**P**eople **L**iberation **A**rmy

Acronyms	Description
PLC	**P**rogrammable **L**ogic **C**ontrollers
R	**R**ecall
RAID	**R**esearch in **A**ttacks **I**ntrusions and **D**efenses
R2L	**R**oot to **L**ocal
RAT	**R**emote **A**ccess **T**ools
RBF	**R**adial **B**asis **F**unction
RGU	**R**obert **G**ordon **U**niversity
RNN	**R**ecurrent **N**eural **N**etwork
ROC	**R**eceiver **O**perating **C**haracteristic
RR	**R**esource **R**ecords
RTU	**R**emote **T**erminal **U**nit
SCADA	**S**upervisory **C**ontrol and **D**ata **A**cquisition **T**echnique
SDS	**S**can **D**etection **S**ystem
SIEM	**S**ecurity **I**nformation and **E**vent **M**anagement
SHA	**S**ecure **H**ash **A**lgorithm
SSL	**S**ecure **S**ockets **L**ayer
SMOTE	**S**ynthetic **M**inority **O**versampling **T**echnique
t-SNE	**T- S**tochastic **N**eighbourhood **E**mbedding
TA	**T**raditional **T**hreat
TN	**T**rue **N**egative
Tor	**T**he **O**nion **R**outer
TP	**T**rue **P**ositive
TPR	**T**rue **P**ositive **R**ate
TTL	**T**ime **t**o **L**ive
TTP	**T**actic **T**echnique **P**rocedure
TTPs	**T**actics **T**echniques **P**rocedures
U2R	**U**ser to **R**oot
UNSW-NB15	**U**niversity of **N**ew **S**outh **W**ales **N**ew **B**enchmark **D**ataset
URL	**U**niform **R**esource **L**ocator
VFDs	**V**ariable **F**requency **D**rives
VPN	**V**irtual **P**rivate **N**etwork
W-IDPS	**W**ireless-**b**ased **I**DPS **T**echnology
-ve	**P**ositive

List of Symbols

***

Symbol	Name	Unit / Formula
AUC	area under the curve	$\frac{1+TP_{rate}-FP_{rate}}{2}$
&	and	
σ	rho	$\sqrt{\frac{\sum(x-\mu)^2}{n}}$
b	bias	
μ	mean	$\frac{1}{n}\sum_{i=1}^{n} x_i$
{	left brace	
}	right brace	
$\in$	is member of	
ω	calculated weight	
β^2	weight parameter	
$\odot$	pointwise or element-wise multiplication	
∂	a partial derivative	
e	euler's number	
$f1$	$f1-score$	$\frac{(1+\beta^2).P.R}{\beta^2 P+R}$
$f()$	non-linear activation function	
f_θ	recursive function	
θ	tunable parameter	
$f(x)$	sigmoid	$\frac{1}{1+e^{-x}}$
x	given attribute	
L	loss function	
P	precision	$\frac{TP}{(TP+FP)}$
$OaAcc$	Over all accuracy	$\frac{(TP+TN)}{(TP+TN+FP+FN)}$
R	recall	$\frac{TP}{(TP+FN)}$
$RELU$	rectified Linear Unit	$ReLU = max(0;x))$
sf	softmax	$\frac{e^{x_t}}{\sum_{j=1}^{n} e^{x_J}}$
s	hidden layer	

Symbol	Name	Unit / Formula
$tanh$	hyperbolic tangent	$\frac{e^{2x}-1}{e^{2x}+1}$
t	current time step	
$\vee$	Vee, denotes logical sum	
$\wedge$	Wedge, and denotes related or dual operators	
Z	ZScore	$\frac{x-\mu}{\sigma}$

Dedication

I dedicate this work to God Almighty who kept me alive and gave me the strength to complete this study. I will not forget in a hurry all the provisions that he made available for me and my kids throughout the period of this study.

Repository note: this page intentionally left blank.

Chapter 1

Introduction

***

One step at a time, diligence,
persistence and focus are keys to
achieving one's goal
-Hope Nkiruka Eke

1.1 Overview of Cyber Security in Current Era 1
1.2 Research Motivation 3
1.3 Problem Statement 4
1.4 Research Aims 4
1.5 Contributions 6
1.6 Thesis Outline 9
1.7 Conclusion 9

In this chapter, an overview of advanced persistent threat (APT) attack, research motivation, aims, questions, objectives, contributions, and outline of the remainder of this thesis are provided.

1.1 Overview of Cyber Security in Current Era

With the constant development of information systems, security of information system remains one of the biggest global concerns. Recorded cases of successful cyber attacks and its methods of operation are also expanding. The attackers are very resourceful, highly skilled, and are being motivated by political or economic interest, espionage and sabotage [1]. Approaches towards defending and protecting critical infrastructures and information systems also need to adapt to these new evolving cyber attacks known as APTs.

Cyber attack has rapidly increased with the evolution of internet for delivering applications and information sharing platform. Some critical infrastructures (CIs), such as industrial control system (ICS) were air-gapped (that is, usually not connected to internet or cooperate IT networks) [2]. The CIs usually operated as separate networks without being connected to public communication infrastructures or platform are now connected to the internet [3]. This interconnection of ICS architectures to diverse communication platforms has benefited businesses by enabling ICS devices to utilize different communication platforms and protocols to increase production efficiency, reduce operational costs

and further improve organization's support [4, 5]. All these improves services supplied to consumers and operators [3, 6].

Thus, previously isolated systems such as distributed control system (DCS) and SCADA, which contains control systems used in nuclear power plants [7], water and sewage systems, and irrigation systems [8], are not immune to or protected by from attack are now connected to internet communication platform. This exposes these systems to variety of threats [9, 10]. The ICS devices such as SCADA system and it's operational protocols are not originally designed or built to support security features such as data encryption or access control, and may often support remote access through radio modems. [11]. Figure [1.1] is a representation of a simple SCADA system. Any potential threat can affect CIs; hence security of each component is extremely important to avoid compromises on any ICS components [5, 12]. Compromise to these systems can lead not only to disruption of operation and huge financial loses but, more importantly, the risk to the public safety.

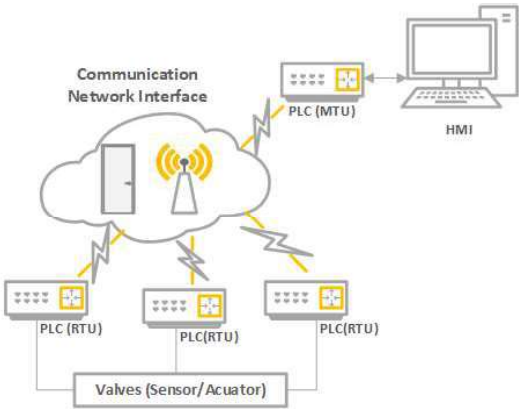


FIGURE 1.1: An example of a simple SCADA system

The cyber threat to CIs such as water plants' operating systems have continued to grow exponentially with respect to the precedence given to the water and wastewater sector, making the sector unsafe. This account for the increase in the number of reported cases of 78.6% since 2014 [13]. The immense public outcry, local water authorities' reactions and regulatory institutions' responses over the case of Oldsmar water attack on February 5, 2021 which exposed over 15,000 residents of Florida's west coast to potential poisoning as a result of unknown remote hackers' activities [14], has caused the disclosure of previously unreported attacks. The hacker used water treatment software to increase the, concentration amount of sodium hydroxide from 100 parts per million to 11,100 parts per million. The concentration was changed immediately by the plant operator back to the correct amount and alerted his supervisor of the incident [14]. The Florida water treatment facility attack has highlighted how some CIs are vulnerable to hacking as a result of their connection to internet, and through the use of remote access programs. In this type of scenarios, implementation of a Zero Trust approach to remote access on operational technology can make a huge difference. This approach uses unique identities and credentials for users and apps to secure CIs. The authors in [2], also highlighted how their findings suggested an "increase in the frequency, diversity, and complexity of cyber threats to the water sector".

Hence, the need to undertake research to understand the best approach towards protecting operational assets from persistent cyber attacks known as APT. Dedicated research to evaluate APT is identified as an important step for ensuring that the company's control systems are safe from cyber attacks.

1.2 Research Motivation

There is a growing concern over the increase in the growth and cost of cyber attacks. The MITRE ATT&CK author in [15] recorded 219 attack techniques which has been released to the public extending the focus from Windows to Linux and Mac platform with complementary knowledge based. Also, in 2015, Morgan et al. [16] reported that the cost of cyber attack is growing at an alarming rate of annual $3 trillion and predicted an increase to $6 trillion, with a cumulative global spending of $1 trillion on cyber security products and services meant for safeguarding against cyber crime between 2017 to 2021. In 2018, the direct gross domestic product (GDP) cost of cyber crime was established as $275 billion to $6.6 trillion globally and total GDP direct plus systemic cost of $799 billion to $22.5 trillion which is between 1.1 to 32.4 percent of GDP [17].

The UK's Office for National Statistics reported an estimation of 4.5 million cybercrimes victims among citizens of England and Wales in 2018. 1.2m were victims of computer misuse, while 3.2m are mainly fraud related. The UK Governments' National Cybersecurity Strategy 2016–2021 recognised that their citizens are insufficiently cyber aware and lacks the skills and knowledge to meet the cybersecurity needs [18]. As reported in [19], this cyber awareness campaign and skill acquisition cost to UK Government is £12m, this equate to about £6 per website visitor.

This high cost has resulted to much interest in investing in research towards developing cyber defence approach. Thus, the national governments recognised the need to create digitally secure environment through cyber aware campaigns for citizens with the goal of improving UK citizens' cybersecurity awareness and skills [19]. However, the main goal of implementing security system measure in most organization is determined and driven by their business objectives and values to protect the confidentiality, integrity, and availability of critical information assets [20].

Cyber threats and malicious activities can be managed through various existing approaches such as (i) static approach; examples are software updates, firewalls, and (ii) dynamic approach; such as intrusion detection and protection systems (IDPSs) which has become a prominent method in identifying diverse malicious foreseen threats. The IDPSs is the most actively studied area since the 1980s [21]. The IDPSs take an active role at the edge of every organisational network to detect any malicious attempt made to gain unauthorized access to network system by the perpetrators. Authors in [22], mentioned that there are no known single practicable IDPSs technique to safeguard systems and CII against APT and other cyber attacks. The type of attack uses unknown vulnerabilities and/or different types of techniques such as social engineering to penetrate their targeted organisation, persistently until their goals are achieved. Also, authors in [23] highlighted that most of the numerous commercially available IDPSs are relatively ineffective and insufficient.

The use of deep learning (DL) based approach has demonstrated a noticeable performance present in the area of automated system for sorting items [24], in health care system automation [25, 26], symbols detection [27], data augmentation [28, 29, 30], face recognition [31], estimation of age and gender classification [32], and attack detection in many practical real life situations. A considerable research interest in the area of implementing an approach based on DL for detecting different stages of APT lifecycle as used by attackers to accomplish an APT full scenario within a system. APT attacks are one of the most dangerous global threats to the multinational corporations companies & Governments [33], and critical infrastructure [2].

DL techniques is not something new. Various IDSPs based on DL have been successfully applied for specific attack of interests detection [34], and anomaly detection [35, 36, 37]. While attackers on the other hand have also advanced in their tactics and techniques. The main challenge lies around detecting multiple techniques used at different stages during APT campaign lifecycle. A technique that will be able to detect any potential presence of anomalies within a system at different process control levels is required. This is to initiate and generate reliable and reasonably accurately detect attack at its earliest possible time in order to either completely avoid system failure as a result of attack, or at least mitigate attack impacts.

Majority of the research work in the area of APTs detection as explored in Chapter [3], focused on detecting only one step of APT lifecycle [34, 38, 39], requires significant expert knowledge for setting up and maintenance [40], can easily be evaded when the infected hosts connect to the C&C domains while users are surfing the Internet [41], depends on an agent to detect elementary attack and has high false positive rate [42]. While some work has tried to detect APT attacks through DL detection approach based on cyber kill-chain model [43], distributed attack detection scheme [44, 45], game theory [46], a provenance graph-based framework [47], deep belief network & support vector machine (SVM) [48], and domain adaptation for windows[49].

The authors in [50] proposed eXtreme gradient boosting and analysis of variance feature selection ML approach and applied it to dataset that contains APT lifecycle. They compared this approach to random forest, decision tree, and K-nearest neighbor and achieved 99.89% accuracy using only 12 features of the dataset. Thus, the current APTs best practices require a wide range of security countermeasures, resulting in a multi-staged defence approach that opens new research directions worthy of study. Hence the need to research new approaches and techniques to safeguard against APT attacks.

1.3 Problem Statement

The problem addressed by this research is that detecting a single step of APT lifecycle does not infer detection of a complete APT full scenario. Considering the complexity of APT attack style, in that APT attack is the type of malicious attack that are carried out in multiple stages or steps using different tactics, techniques and procedure (TTPs) at different stage to deliver and exploits the network and resources of their target. This study investigates the problem and proposes a solution to address these multi-steps attack campaign of APT.

1.4 Research Aims

This research aims to investigate APT attack detection and develop a novel multi-step APT attack detection framework to address detection of multi-steps of APT campaign. When this framework is implemented, it will accurately detect APT attack at its earliest possible stage, contributing to intrusion detection systems. The ability of the proposed approach to detect an APT attack should be high with low false alert as to determine the performance and effectiveness of the approach. Accordingly, the following research questions *(RQ1-4)* in Sub-section [1.4.1] are set as guidelines to realize this research aim.

1.4.1 Research Questions

This research seeks to investigate the following research questions based on the study interest:

(*RQ*1) What are the main features that constitute an APT lifecycle in terms of individual step - TTP, used during each stage of APT campaign?

Understanding the main features that constitute APT attack steps will support building a system that can detect or defend against APT attacks. This leads to second research question *(RQ2)*.

(*RQ*2) What measures could be used to build an effective active defence against APT attack in a holistic way?

The ability of a system to detect or mitigate an attack, the detection system should be able to detect an attack quickly and early enough. This is to reduce the extent of damage that may have occurred if the attack was successfully executed. It will also prevent further break-ins by attackers. This leads to the third research question *(RQ3)*.

(*RQ*3) How can developing and implementing a multi-stage model based on DL techniques help to implement an effective APTs detection system?

Implement a system that can detect the first step of an APT attack, thereby preventing the execution of other steps that may lead to full APT scenario, thus pre-empt attackers from accomplishing their goals. This leads to the fourth research question *(RQ4)*.

(*RQ*4) How good and effective is the developed system in terms of detection capability?

Choosing the right algorithm and parameters, and fine tuning the developed model's optimal hyperparameter during training has an impact on how good the developed model will perform. Also using the right metrics suitable for ascertaining the performance of this model in terms of attack steps detection in support of research question *(RQ3)*.

1.4.2 Research Objectives

The main objectives of this study are to answer the research questions *(RQ1-RQ4)*, as listed in Sub-section [1.4.1] using the following research objectives *(RO1-RO3)* as guidelines.

(*RO*1) Undertake an in-depth study of APT lifecycle / traits, and investigation of APT with respect to multi tactics and techniques used to deliver APT attacks. This objective addresses research question *(RQ1)* in part.

(*RO*2) Investigate the state of the art of diverse approaches for APTs detection and prediction. Design, develop, and build an experimental framework to compare / contrast various methods.

A good understanding of the existing APT detection methods, will serve as a good starting point to know what has been done in this area. It will also help to develop a framework for detecting APT attacks. Application of the developed model to an evolving system for defending the system operation of gas pipeline and to detect APT attack at earliest possible steps before completing its lifecycle will make great contribution to system security. Part of *(RQ2)* and *(RQ3)* are addressed within this objective.

(*RO*3) Experiment and evaluate the developed framework in different problem domains datasets, so as to ascertain how effective and good the framework performed. Using a historical record of monitored network to evaluate the performance of the system in terms of detecting each individual APT attack step. The research questions *(RQ2)*, *(RQ3)* and *(RQ4)* are addressed within this objective in part.

1.5 Contributions

Targeted APT attacks have presented an increasing concern to both government and business continuity. Detecting such targeted and rare attack is difficult as it is a well planned attack, executed through low and slow approach using combination of different sophisticated techniques. This approach gives rise to fewer network events records that are associated with APT scenario, resulting to classification issue with imbalance data [1]. Considering that execution of full APT attack scenario occurs in diverse steps, an approach that will deal with unevenly distribution of rare attack events among other attack steps focusing on capturing each individual step at different system structural level as early as possible is highly recommended. This thesis argues that a model based on DL that utilizes ensemble techniques (for optimizing detection accuracy by combining network results) on recurrent neural network (RNN) variants due to the capabilities of RNN variants to:

- ☐ *constantly learn dynamic behaviour over a time sequence data*
- ☐ *captures information from sequences of time series data about change over time*
- ☐ *detect & respond,*
- ☐ *and works well with time series data could make a good model for APT attack steps detection.*

This DL-based model if configured with the appropriate hyperparameter settings such as activation function, optimisation function, loss function, and correct use of data pre-processing (to deal with missing and imbalance data) is the right path to achieve an effective neural network model, capable of dealing with this type of attack that are been executed in different stages in a real-time dynamic problem environment. Hence, the main contributions in this thesis are in the domain of cyber security. These contributions are listed as follows:

- To answer the first research question ($RQ1$), an extensive investigation was carried out to understand APT mode of operation and to identify current state-of-the-art techniques geared towards detecting APT attacks. The current position of publicly available APT attack datasets for research purposes was studied as well as the APT characteristics, and lifecycle. Examples of the most significant confirmed cases of APT attack on CPS devices were also given. Most of the reviewed detection approaches focus on dealing with specific threat and can only detect one step of APT lifecycle. Detection of a single APT step does not indicate detection of APT scenario as APT actors uses several TTPs to achieve their goal. This study believes that detecting an APT should involve keeping track of different steps taken over the period and linking the relationship between the steps triggered during APT lifecycle. Hence, this study has identified the need to develop a multi-stage detection framework for APT lifecycle detection. In line with research objective ($RO1$-$RO2$) in Sub-Section [1.4.2]. The outcome of this part of study was published as a book chapter in *"Security and Resilience in Cyber-Physical systems, Detection, Estimation and Control"* [5].

- Design of multi-stage attack detection system (see: Figure [4.1] in Chapter [4]). This study proposed a novel framework based on *"deep APT steps analysis and correlation of APT lifecycle"* abbreviated as ***"APT$_{DASAC}$"***. The framework utilizes ensemble deep neural networks (DNNs) for realising multi-stage security detection system. It takes

RNNs variants to learn features from raw data, captures the malicious sequence patterns which reduce the cost of artificial feature engineering. Preliminary critical investigation of DL-based methods for attacks detection was carried out to determine the potential of developing ML /or DL-based models for APT detection. Build an experimental framework to compare and contrast various traditional methods [51]. Explored applications of data resampling techniques, together with heterogeneous ensemble approach for managing data imbalance caused by unevenly distributed data elements among classes with focus on capturing the rare attack. The overall achieved result suggests that high classification error and class separability problem of minority class has a noticeable impact on model overall performance with respect to application domain as seen with the case of KDDCup99 and UNSW-NB15 datasets. [1]. Further discussion on these preliminary investigations can be found in Sub-section [3.6]. This contribution took care of research question ($RQ2$).

- To address research question ($RQ3$), a novel multi-stage framework that utilized historical record of the monitored network, and applies DL-based method for attack detection has been developed. I designed and implemented a multi-stage security detection DL-based approach termed ***"APT$_{DASAC}$"***. This approach takes into consideration the distributed and multi-level nature of ICS architecture and reflects on the four main SCADA-based cyber attacks [52]. Furthermore, stacked ensemble for APT_{DASAC} to combine networks' results for optimizing detection accuracy is utilized. The implementation of this designed framework for APT attack detection system is built to run through three stages as follows:

  - *Stage 1: Data Input and Probing Layer*
  - *Stage 2: Analysis & correlation Layer*
  - *Stage 3: Decision Layer*

  Chapter [4] contains the detailed description of this developed framework.

- Application of the developed framework were implemented to address research question ($RQ4$). Series of experimental evaluation, to analyse the performance of the developed approach in four different domains. This includes, application to the new gas pipeline (NGP) [53] dataset collected by simulating real attacks and operator's activity on a gas pipeline. Detection of individual APT attack step and attack type classification were conducted. The achieved results suggest that the implemented approach has shown attack detection capability, and has demonstrated that performance of attack detection techniques applied can be influenced by the nature of network transactions with respect to the domain of application [11].

The design and developed multi-stage framework with generic settings that will consider different level of system architectural design (for this study - SCADA system) and handle APT as a multi-step attack, which can also be deployed in different domain and achieve the same purpose of safeguarding a system against any malicious intrusion such as APT is a great contribution to cyber security community. The findings also, suggests that a hybrid of model based on "DL" & "ML" approaches with focus on capturing every single detectable step of APT lifecycle will make a good model for APT attacks detection. This is to take advantage of the combined effort of both models, to achieve a more effective detection capability since both approaches demonstrated significant attack detection capability.

1.5.1 Research Outputs Links

In this section all the publications which have arisen from the research are listed. Available at;

- **RGU Worktribe:**
  https://rgu-repository.worktribe.com/person/871584/hope-nkiruka-eke/outputs
- **Google Scholar:**
  https://https://scholar.google.com/citations?user=t5XO0rUAAAAJ&hl=en&oi=ao

1.5.1.1 List of Publications

This thesis contributions and results have been published in the following journals and conference proceedings, as well as a book chapter:

- **Conferences:**
  - Eke, Hope Nkiruka., & Petrovski, Andrei. (2023, May). Advanced Persistent Threats Detection based on Deep Learning Approach. In 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS) (pp. 1-10). IEEE. Available from: https://doi.org/10.1109/ICPS58381.2023.10128062
  - Eke, Hope, Andrei Petrovski, and Hatem Ahriz (2020a). "Detection of false command and response injection attacks for cyber physical systems security and resilience". In: 13th International Conference on Security of Information and Networks, pp 1-8. Available from: https://doi.org/10.1145/3433174.3433615
  - Eke, Hope Nkiruka, Andrei Petrovski, and Hatem Ahriz (2019). "The use of machine learning algorithms for detecting advanced persistent threats". In: Proceedings of the 12th International Conference on Security of Information and Networks, pp. 1-8. Available from: https://doi.org/10.1145/3357613.3357618
- **Book Chapter:**
  - Eke, Hope Nkiruka, Andrei Petrovski, Hatem Ahriz, and M. Omar Al-Kadri (2022). "Framework for Detecting APTs Based on Steps Analysis and Correlation." In Security and Resilience in Cyber-Physical Systems, pp. 119-147. Springer, Cham, 2022. Available from: https://doi.org/10.1007/978-3-030-97166-3_6
- **Journals:**
  - Eke, Hope, Andrei Petrovski, Hatem Ahriz (2020b). "Handling minority class problem in threats detection based on heterogeneous ensemble learning approach". In: International Journal of Systems and Software Security and Protection (IJSSSP) 11(2), pp. 13-37. Available from: https://doi.org/10.4018/IJSSSP.2020070102
- **Symposium:**
  - Eke, Hope Nkiruka, Andrei Petrovski, and Hatem Ahriz (2019). "SYMPOSIUM: Securing Information Systems against Advanced Persistent Threats (APTs)". The Graduate School Symposium. Graduate School Symposium 19 June 2019.

- **Poster:**
  - Eke, Hope Nkiruka, Andrei Petrovski, and Hatem Ahriz (2018). "POSTER: Securing Information Systems in Oil and Gas Industry against Advanced Persistent Threat (APT)". The Scottish Informatics & Computer Science Alliance (SICSA) PhD Conference 28-29 June 2018. The recorded video can be accessed from https://www.youtube.com/watch?v=tSdoHhcLoXo

1.6 Thesis Outline

The remainder of the thesis is organized as follows:

Chapter [2] provides the review of literature on APT, it's lifecycle, intent, and types. Cybersecurity countermeasures & mitigation are also provided. Besides, ICS cyber security objectives, threats, issues, and design tools, the intrusion detection and prevention systems (IDPSs) were also explored. In addition, DNNs and machine learning (ML) approaches relevant to the study were presented.

Chapter [3] explores the existing state of the art for the APT attack detection, prevention, and mitigation. This chapter also introduces some of these research findings on APTs detection while highlighting some of the limitations of these existing approaches. The ATT&CK Matrix TTPs and some helpful network traffic attributes that can be used to uncover malicious activities were also presented. The four datasets used for this thesis, it's attack types and representations as aligned to the study were also discussed. In addition, preliminary experiments carried out over the course of this study are presented.

Chapter [4] presents the detailed description of the three main phases of the designed approach "architectural design of APT_{DASAC}". APT_{DASAC} - a deep learning multi-staged security detection based approach, that takes into consideration the distributed and multi-level nature of ICS architecture and reflect on the four main SCADA-based cyber attacks and APT lifecycle was detailed.

Chapter [5] explains the implementation of APT_{DASAC}, highlighting all frameworks, tools and programming languages used for this implementation. The implementation pseudocodes and algorithms of each individual phase of APT_{DASAC}: data input layer, data analysis layer and decision layer were presented separately.

Chapter [6] discussed the evaluation of APT_{DASAC} and presents the achieved result. In addition, a comparison between other existing systems to the thesis achieved result and the developed system APT_{DASAC} is provided.

Chapter [7] provides an overall conclusion of the thesis, summarises the achievements and findings. Outline of the limitations of the work presented in this thesis, and also some suggestions for possible further research work are summarized. References and appendices were documented after Chapter [7].

1.7 Conclusion

This chapter has given a brief background to the research, describing the motivation behind the study, stated the problem statement and also highlighted the research questions addressed. It has also briefly set out the research aims and objectives. The original contributions has been highlighted and the thesis structure outlined. Finally, the outputs which has arisen from this research, which includes the list of publications and presentations made in connection with this research has also been included in this chapter.

Thesis written by —
Hope Nkiruka Eke: Research Output
Google Scholar: Google Scholar
RGU: University Website

Chapter 2

Background and Related Work

***

2.1 Advanced Persistent Threats (APTs) 11
2.2 Cybersecurity Countermeasures & Mitigation 17
2.3 Industrial Control System (ICS) 18
2.4 Intrusion Detection and Prevention Systems (IDPSs) 28
2.5 Deep Neural Network (DNN) 37

In this chapter, an overview of APT lifecycle, the goal of APT attacker, few examples of successful attacks, and few cybersecurity countermeasures & mitigation are provided. Also, discussed ICS cyber security objectives, threats, issues, and available useful tools for designing security measures. The IDPSs are also explored. In addition, DNNs and ML approaches relevant to the thesis are presented.

2.1 Advanced Persistent Threats (APTs)

Cyber attack such APT is continuously evolving, posing a devastating security risk for organizations and governments alike. Available IDPSs mechanisms are not able to tackle this type of attack [54]. Understanding APT constituent and its mode of operation will help towards continuous development of approaches that will protect systems or/ minimize APT attack impact.

2.1.1 Defining APT

There are diverse views as to what makes a threat an APT, as summarized in Figure [2.1]. Some authors believe that an APT is a nation-state sponsored attack [55, 56], as a term that is frequently been used in security threat discussions [57], while [58, 59] offer the following definition: "APT is often aimed at the theft of intellectual property or espionage as opposed to achieving immediate financial gain and are prolonged, stealthy attacks". However, work in [60, 61, 62] view APTs as a highly sophisticated combination of different techniques to achieve a specifically targeted and highly valuable goal.

Although there are various definitions of APTs, the US National Institute of Standards and Technology (NIST) [63], states that an APT is "an adversary that possesses sophisticated levels of expertise and significant resources which allow it to create opportunities to achieve its objectives by using multiple attack vectors (e.g., cyber, physical, and deception). These objectives typically include establishing and extending footholds within the information technology (IT) infrastructure of the targeted organizations for purposes of

exfiltrating information, undermining or impeding critical aspects of a mission, program, or organization; or positioning itself to carry out these objectives in the future". According to NIST, the key features of an APT are: "(i) pursues its objectives repeatedly over an extended period of time; (ii) adapts to defenders' efforts to resist it; and (iii) is determined to maintain the level of interaction needed to execute its objectives".

Interestingly, author in [55], constructed three basis by which an attack is not an APT: (i) any successful attack that may have been prevented through so many means (assuming that the attack occurrence did not come as a surprise and is relative to that domain). (ii) secondly, if an attack was delivered and achieved their goal with less intense techniques and (iii) thirdly, an attack that did not exhibit any new techniques should have been detected by those available tools and techniques.

Nevertheless, this study defined APT as a well planned attack, executed through low and slow approach using combination of different sophisticated techniques by very resourceful and highly skilled organized attackers, who are being motivated by specific goals (which could be but not limited to political or economic interest and espionage) against their target.

APTs, which were originally used to describe cyber intrusions against military organizations, are not limited to the military domain [64, 65]. As highlighted in several large-scale security breaches [65, 66, 67, 68, 69, 70]. There have been quite a few targeted attacks in the energy sector, these are ***Stuxnet, Duqu, Shamoon/Disttrack and Night Dragon*** while ***Hidden Lynx, Nitro, Flamer, Net Traveler and Elderwood*** are targeted towards power companies. Also, the ***Oldsmar water treatment plant***, that targeted Florida water treatment plant [13, 14]. One of the most prominent examples, and a game changer for many organizations, was Stuxnet. This targeted sabotage attack, which was aimed against uranium enrichment facilities in Iran, made clear what could be done through cyber attacks in ICS.

A new development, such as further extensions of smart grids and smart metering, expose more CI to the internet. Critical infrastructures that are segregated from the internet and other networks are not immune to today's evolving threats as demonstrated by the Stuxnet attack [52]. This attack primarily targeted SCADA control systems for industrial power plants. Operators of Essential Services (OES), as well as energy utility companies, need to be aware of these threats and prepare accordingly. A comprehensive and clear understanding of the APTs lifecycle together with a reliable detection system that has been tested on a real-time APT scenario is an ongoing challenge.

2.1.2 APTs Traits

APTs and the actors behind them constitute a serious global threat. This type of attacks differs from commodity threats that seek to gain immediate advantage and are broad in their targeting and process [51]. APT on the other hand is very;

- ☐ *resourceful*
- ☐ *with well defined objectives and purpose*
- ☐ *uses sophisticated methods and technology - uses different TTPs*
- ☐ *substantially funded.*

The APT threat process follows a staged approach to find, penetrate and exploit their target. Understanding the ***advanced***, sophisticated tactics, techniques and ***persistent*** nature of APT is unavoidable in defending against an APT ***threat*** (see Figure [2.1]).

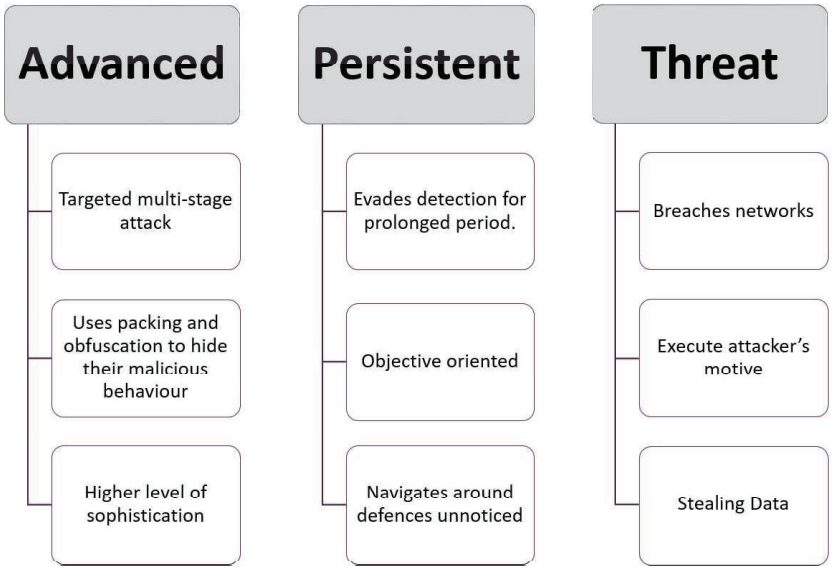


FIGURE 2.1: Advanced Persistent Threat (APT)

- ***Advanced*** - The advanced nature of APT provides the attackers with the capability of maintaining prolonged existence through stealthy approach inside an organization once they successfully breach security controls. Attackers uses sophisticated tools and techniques such as malware, if the malware is detected and removed, they change their tactics to secondary attack strategies as necessary. They look for other ways around any internal security controls in the event they lose their initial established foothold [40].

- ***Persistent*** - The meaning of "persistent" is expanded to the acts of the attackers persistently launching spear-phishing attacks against their targets. The attackers navigate the victim's networks from system to system, obtain information, monitor network activity, and adapt to be resilient against new security measures while maintaining a stealthy approach to reach the targets [71]. The mode of attack indicates the main functions of the APT-type malware, usually focusing more heavily on spying instead of financial gain.

- ***Threat*** - The threat actors also have the capability of gaining access to electronically stored sensitive information [58]. Other than the purpose of collecting of national secrets or political espionage, based on the functions discovered, it is believed that this threat can also apply to the cases in business or industrial espionage, spying acts or even un-ethical detective investigations [5, 72, 73].

2.1.3 An Overview of APT Lifecycle

APT attacks are generally known to utilise a zero day exploit of unpublished vulnerabilities in computer programs or operating systems together with social engineering techniques to maximise the effectiveness of the exploits [58]. Launching an APT involves numerous hacking tools, a sophisticated pattern, high level knowledge, varieties of resources and processes as an APT attack is not a single step attack unlike other attacks [74]. This suggest the need to develop a multi-stage detection framework for APT lifecycle detection. This approach will be keeping track of different steps taken over the period and linking the relationship between the steps triggered during APT lifecycle. APT proved extremely effective at infiltrating their targets, going undetected for extended periods of time, increasing

their appeal to hackers who target businesses as highlighted in several large-scale security breaches [68, 70, 75].

Each APT attack is customised with respect to attacker's target and aim at each stage. The patterns of APT attacks are similar in most cases but differ in the techniques used at each stage to deliver. The six basic APT attack phases as shown in Figure [2.2], based on literature review in combination with the "Intrusion Kill Chain (IKC)" model described in [40, 76, 77] are as follows.

- ***Reconnaissance and Weaponisation:*** This stage involves information gathering about the target [11, 78]. This could be, but not limited to, organizational environment, employees' personal details, the type of network and defence measure in use. This information gathering can be carried out through social engineering techniques, port scanning and open source intelligence (OSINT) tools [11].

- ***Delivery:*** At this stage, attackers utilise the information gathered from reconnaissance stage to execute their exploits either directly or indirectly to the targets. In direct delivery, the attackers use social engineering such as spear phishing by sending phishing email to the target while in indirect delivery, attacker will first compromise a trusted third party, which could be a vendor or frequently visited website by target, and uses these to deliver an exploit [79].

- ***Initial Intrusion and Exploitation:*** In the intrusion stage, attacker gain access to target's network by utilising the credentials gathered through social engineering. The delivered malware code is downloaded, installed and activated backdoor malware, creating a command and control (C&C) connection that links the target and the remote attacker's system [80, 81, 82]. Once an attacker has secured connection to the target system, attacker continues to gather more security related information such as security configuration and user names, sniffs password from the target network while maintaining a stealthy behaviour in preparation for the next attack.

- ***Lateral Movement and Operation:*** At this stage, once the attacker has established communication between the target's compromised systems and servers, the attacker moves horizontally within the target network, identifying the servers storing the sensitive information of users with high access privileges. This is to elevate their privileges to access the sensitive data, making their activities undetectable or even untraceable due to the level of access they have [79, 83].

- ***Data Collection:*** This stage involves utilising the privileged users' credentials captured at the previous stage to gain access to the targeted sensitive data. With the attackers having a privileged access, they will now create redundant copies of C&C channels using sophisticated tools should there be any change in security configuration [84, 85, 86, 87, 88]. Once the target information has been accessed, redundant copies are created at several staging points, where the gathered information is packaged and encrypted before exfiltration [84, 89].

- ***Data Exfiltration:*** At this stage, once an attacker has gained full control of target systems, they proceed with the theft of intellectual property or other confidential data. Stolen information from a staging server is transferred to attackers external servers either in the form of encrypted packages, password protected zip files or even through clear web mails. The idea of transferring information to multiple servers as drop points is an obfuscation strategy to stop any investigation from discovering the final destination of the stolen data[79, 90, 91, 92].

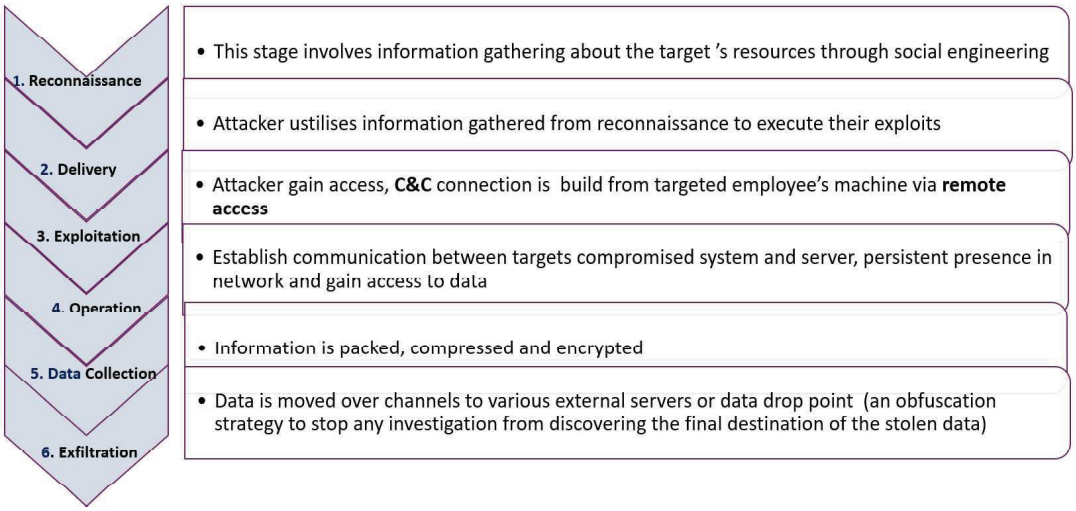


FIGURE 2.2: APT Lifecycle

2.1.4 APT Intent

The APT attack goal usually focuses on but not limited to political interest, trade secrets, espionage, proprietary, sensitive economic and/or national security information [5].

Investigating the APT intention and the new methods being used to breach today's security controls, it emanates to a basic understanding that attackers, especially those who have significant financial motivation, have devised effective attack strategies centered on penetrating some of the most commonly deployed security controls (largely signature-based antivirus and signature-based intrusion prevention), most often by using custom or dynamically generated malware for the initial breach and data gathering phase.

The 'Advanced' and 'Persistent' are two major features that differentiate APT from basic and Traditional Attack (TA) as illustrated in Tables [2.1] and [2.2]. These characteristics made APTs ideal for campaigns against enterprises as the perfect tool that easily penetrates defences, avoids detection for lengthy periods of time and steals sensitive information.

TABLE 2.1: Two Significant ways APT differ from Traditional Threat

Attributes	APT are Persistent	APT are Advanced
Goal	The goal of an APT is the desire to acquire assets from or disrupt their target operation	- Attackers are only interested in breaching specific target irrespective of the cost - Attacks are delivered over multiple vectors and may be crafted around 0-day exploits - Traditional signature based detection are not suitable
Cost	As a result of high cost of executing an APT - high strategic values are targeted - attackers are well funded and organized - large and influential organisation are targeted	- Attacks are customized
Entry Point	- Attackers breach any potential vulnerability they can find instead of using commodity attack - their entry point is social in nature	APT is agent oriented, patching any technology vulnerability is not enough. - Attackers craft new payload for any organisation that remains unhardened.
Mitigation	**APTs should be considered as an actor threat that requires a comprehensive mitigation strategy**	**Traditional cyber security focuses more on technology vulnerability instead of the attacker and such will not work for APTs**

TABLE 2.2: Differences among Basic, Traditional Attack and APT

Attributes	Basic Attack	Traditional Attack	APT Attack
Attackers	Amateur hackers	Extortionists, mature cyber criminals	Nation States, sophisticated adversaries Hacktivist Groups, Group of organised crime
Examples	Generic phishing scams	Distribute Denial of Service (DoS)	Highly sophisticated adversaries who can bypass virtually all of today's "best security practice"
Approach	Generic phishing scams	Developed techniques, usually single-run, "smash and grab" over a brief period	Stealthy repeated attempts, stays low and slow, Adapts to resist defences over lengthy period
Tools	Cyber techniques available on internet open source	Customised tools	Operate through customised exploits, spear phishing, zero-day exploits, remote access tools (RAT)
Target		Unspecified, mostly individual systems, targeted, private data extraction	Specific organizations, governmental institution, commercial enterprises
Purpose		Extortion as motive, financial benefits, demonstrating abilities	Long term, persistent occupancy for strategic benefits such as data thefts, intelligence espionage, and other malicious activities and competitive advantages
Maturity Level			
Maturity Level	Simple, easily accessed tools, done by armature hackers and not particularly targeted.	Technical mature, developed by advanced individual or teams, but not coordinated or extremely targeted.	Sophisticated planned over a lengthy period, complex and targeted

2.1.5 Examples of known APT Attacks

APT attacks have affected many organizations. As far back as 1998, the first public recorded attack named Moonlight Maze targeted Pentagon, National Aeronautics and Space Administration (NASA), the US Energy Department, Research Laboratories and private Universities [93]. This attack successfully compromised Pentagon computer networks and accessed tens of thousands of files [57]. Over the past years, there has been an increase in the number of organizations coming forward, admitting they have been targeted. Unfortunately, in the bid to protect organization image and to avoid providing hackers with feedback, majority of those organization are not willing to share the details of the attacks against them [58].

Nevertheless, few major recorded targeted attacks against ICS are ***Stuxnet, BLACKENERGY 2, HAVEX and CRASHOVERRIDE***. Stuxnet is the first ever recorded attack aimed at disrupting physical industrial processes resulting in violation of system availability, while CRASHOVERRIDE is the second and also the first known to specifically target the electric grid [94, 95]. CRASHOVERRIDE is not unique to any vendor or configuration but utilises the knowledge of grid operations and network communications to cause disruptions leading to electric outages [96, 97].

Deep panda is an attack that was believed to be of Chinese origin, carried out against industries, including US Intelligence Service, government, defense, financial, and telecommunications using Deep panda code to acquire staff information that affected over 4 million employees' information [98, 99].

Epic Turla also known as Snake or Uroburos, targeted government agencies, state departments, military agencies and embassies across over 45 countries worldwide with France at the top of the list. This type of attack used direct spear-phishing e-mails and watering hole attacks to infect their target. It was discovered by Kaspersky [100, 101]. It used Epic backdoor to connect to the C&C server as soon as the target system is infected, then sends a pack with the target system information containing a brief summary of the

compromised target system to the attacker. A pre-configured batch files containing series of commands for execution is delivered and also uploaded custom lateral movement tools (such as RAR archiver) based on the target system information received by the attacker as highlighted in [100].

Another recorded CI attack is the ***Oldsmar water treatment plant*** attack which exposed residents of Florida's west coast to potential poisoning [13, 14]. This attack highlighted how vulnerable some CIs are to hacking as a result of their connection to internet, and the use of remote access programs. Implementation of a Zero Trust approach to remote access on operational technology may help to secure CIs to a certain degree.

These attacks on CIs have highlighted the need for robust and reliable system to be put in place as a security measure for CIs. The need for continuous research in improving the existing method and research new methods and techniques for detecting, mitigating APT attack cannot be over emphasized. Hence, the need to undertake research to understand the best measure towards protecting operational assets from against this dangerous cyber attack called APT. This research is identified as important requirement for ensuring that the company's control systems are safe from this multi-step attack threat.

2.2 Cybersecurity Countermeasures & Mitigation

Fundamentally, it is of essence that the cyber security practitioners understand not only what threats a model capability claims to solve, but also how to address those threat from an engineering perspective, and also understand at what condition any applied mitigation solution will be successful. A few approach has been proposed towards defending against targeted APT attack/or cyber security attack in general so as to reduce the impact of any successful attack at any stage within the targeted system(s).

In this regard, a cybersecurity countermeasure is any approach or technology which could be software or hardware, developed to mitigate or defend against cyber security activities. Thus, to effectively deploy a countermeasure and achieve it's main functionalities, a security architect need to understand the actual problem the solution is trying to resolve, if such system has previously been applied to similar problem, and how the same issue has been dealt with in past if applicable in other to deduce why the proposed solution will be better [102].

For this research framework, the triggered malicious alert email is sent to request tracker [103], where the network security team will streamline the alert to enable the team carry out thorough additional forensics investigation and respond to the triggered alert appropriately. However, any triggered alert within the set time window, say 24 hours, that has the same malicious alert features previously triggered and emailed is suppressed. This alert suppression helps to reduce the computational cost of the alert clustering & correlation module (ACCM).

The authors in [102], created an encoded knowledge base graph countermeasure framework called D3FEND. This framework described how the graph can support queries that construct and map cybersecurity countermeasures to offensive TTPs. This graph is utilized to define the concepts in the cybersecurity countermeasure domain and the relations necessary to link those concepts to a particular reference in the cybersecurity literature, by analyzing sources of research and development literature of over 500 samples of targeted countermeasure patents from the U.S. Patent Office corpus between 2001 to 2018.

The authors [102], also established a semantic model of a portion of MITRE's ATT&CK framework [104], that "represents offensive TTPs with the same common, standardized semantic language and incorporate ATT&CK by mapping its concepts directly to D3FEND's model of defensive techniques and artifacts". This framework result has demonstrated a promising piece of work to cyber security communities. However, further validation of this graph countermeasure framework successful use requirements analysis is necessary.

Again, NIST Cybersecurity Framework created as a set of guidelines, standards and best practice for organizations to better manage their cybersecurity risk by aligning their cybersecurity activities around the five main functions, which are Identify-Protect-Detect-Respond-Recover paradigm [105]. NIST maintains the U.S. government National Vulnerability Database, the repository of standards-based vulnerability management data for security controls enumerated in NIST 800-53. These pertinent security controls has been mapped by NIST to the appropriate activities as defined in the Cybersecurity Framework [106].

Furthermore, attack mitigation is very important when a system is under attack as discussed in Sub-section [3.2.3]. Author in [107], suggested an increase in organizational security maturity level within the system by implementing an automated response system as the workhorse of the rapid response process. Also, antivirus software and the operating system update with the latest patches was proposed in [108], implementation of users' data access restriction based on user that poses behaviours that can initiate APT attach due to risk associated with BYOD in [109], implementation of access & usage policies and controlling external media [110]. Creating awareness and educating system users & system administrators as well as web-based communities users on different attack vectors will equip system users with necessary information and knowledge towards system protection, thus helping to reduce attack impact [110, 111, 112].

In Sub-section [3.2.2], defensive mechanism towards mitigating APT impact were discussed. Examples are - game theory concept for defender to predict future threats [113], two-layer differential game framework [114], hybrid strategy game-based on dynamic defense model to optimally allocate constrained secure resources against APTs in a wireless sensor network [115], & APT defence strategy as an optimal control problem [116] are proposed to help defend against APTs attack. Time is of great value at the face of targeted adverse APTs organised cyber threats, few security proposed countermeasures and mitigation approaches that may help if implemented to reduce the impact of successful attack on a system were also discussed in this Section.

2.3 Industrial Control System (ICS)

ICSs are cyber physical systems that are responsible for maintaining normal industrial plants operation such as gas pipelines, water treatment and power plants. SCADA is a type of ICS that monitors and provides control to remote physical systems through centralised control system in real-time. SCADA systems gather information from remotes facilities about the state of the physical process and send commands to control the physical process creating a feedback control loop communication. The most expensive and fastest growing consequence of cyber crime is information theft as well as core systems, such as ICSs being attacked as mentioned in [117].

The need for robust and resilient ICS security is obvious as these systems are becoming more vulnerable to outsider threats with increased network connectivity. Hence, secure and effective communication protocols that require both time-critically and security in the ICSs' distribution and transmission systems are essential. Availability, safety, integrity,

confidentiality and authentication are the main ICS high-level cyber security objectives [118, 119, 120], represented in Figure [2.3]. However, system availability, safety reliability and integrity are still the most important security objectives in the context of ICS security.

2.3.1 ICS Security Objectives

The main aim of implementing security system measure in any organization is determined and driven by their business objectives and value to protect the confidentiality, integrity and availability of critical information assets. In the case of protecting an ICS devices, the availability of control systems, safety of human life and the integrity of the data that is been processed is of utmost importance [20], as ICS resources are required to be available at all time.

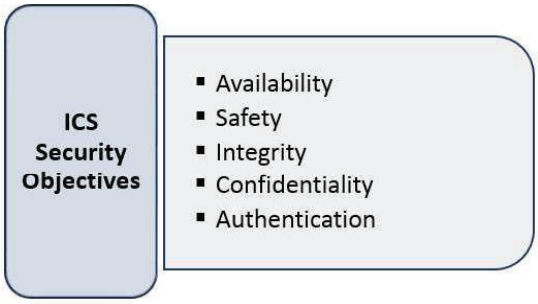


FIGURE 2.3: Security Objectives related to an ICS

- ***Availability:*** Accessibility and availability of any system resources when needed is of paramount importance, especially for ICS operation. This is because being unable to access control information, or any loss of availability, will lead to partial or total unavailability of the system's services, which may be caused by a DoS attack. Therefore high system service availability is required to prevent the system from power outages, communication corruptions due to hardware failures and system upgrades. At the same time, ensuring that the information is only accessible by authorized users and systems.

- ***Safety:*** Ensuring the Cyber safety of an ICS device deals with the protection against various attacks in the cyberspace [121]. Any successful attack on an ICS device may incur a huge impact. These impacts could result in weakening the safety measures, confidentiality, integrity, resources availability, system authentication loss or harm. These could be harmful to users, environment or systems. As a result of the distributed and multi-layered nature of the ICS, users are gradually becoming a critical asset to protect, in addition to other control functions [8].

  Safety issues within the interconnected computer-based system have become extremely important and require a systematic approach to safety assurance, subject to a possibility of unreliable network communication [122]. The architectural design of system safety should be well understood. There exist well established safety standard, IEC 61508, specifying safety integrity levels [123].

- ***Integrity:*** This refers to the system operation, how it is intended to function without being degenerated through modification of information sent and received by the sensors, actuators, or controllers [124]. Integrity involves the system's ability to prevent the adversary, who intentionally modifies or deletes important data, often resulting in false data being transferred [118], i.e., ensuring the authenticity of information.

- ***Confidentiality:*** Refers to the ability to protect information from being accessed by unauthorized users or systems. A breach of confidentiality means that an unauthorised user or system has gained access to sensitive system information. Adversary can achieve this on their target system through eavesdropping and intercepting the communication channels between the controller and the actuator or sensors[125]. Hence, confidentiality is necessary for maintaining the users' privacy [118].

- ***Authenticity:*** Authenticity is the process of ascertaining the identity of a system user in computing and communication process to ensure data, transactions, communications are legitimate to eliminate any form of attacks that may be targeting system data integrity [119]. The basic requirements for system authentication protocol, needed to ensure reliable security to protect data integrity, are as follows:

  - *tolerance to faults and attacks*
  - *high efficiency and*
  - *support of multicast*

  The comprehensive description of these authentication protocol can be found in [119].

2.3.2 Security Issues

Cyber security is believed to be one of the key global security issues of our times [126]. The focus of security issue are information attacks (either direct or indirect), political espionage and un-ethical detective investigations etc. An advanced intruders' first goal is to gain full control of their target system, as to have unrestricted access to the inner functionality of the system. This will enable the attacker to achieve their main goal.

Cyber security challenges affect different areas of life, ranging from individual, businesses, and governments. It is also challenging for students and academics in general as well as, organizational critical infrastructure. Many critical components of ICS such as controller (eg. PLC), control network, supervisory software designed without security in mind, are vulnerable to cyber attacks [127]. Successful compromise on any of these components, will not only disrupt organizational operation, cause huge financial loses, total system collapse, and more importantly, pose a big risk to public safety.

- **Examples of Security Challenges:**

  - PLC is designed to withstand industry's harsh operating environment. It collects data and interacts with sensors, valves, motors and other devices deployed within the industrial systems for effective system process automation and control [128]. Any attack on PLC has serious impact.
  - PLCs usually run for several years, firmware patch or update is inevitable for bugs fix or to add features to PLC. Firmware modification attack whereby an attacker is able to create and upload malicious firmware to the controller will leave PLC under malicious control [127, 129].

There exist several ongoing dedicated research work including this study, in search of better way to protect or minimise the impact of successful attack on CIs devices. Thus, security of each component unit is extremely important to avoid compromises and be in a better position to achieve ICS security objectives, which are system availability, safety, integrity, confidentiality and authentication (see Sub-section [2.3.1] and Figure [2.3]).

2.3.3 Stealth Cyber Attacks

Apparently, threats to network security remain one of the biggest challenges facing organizations and industries at various levels of operation. Cyber attacks are the most dangerous threats that CIIs are facing among other CIIs challenges [130]. Actors can directly, or indirectly through remote access, execute malicious exploits that may have serious impacts on the victim's infrastructure.

The severity of these attacks are increasing, the ICS cyber emergency response team (ICS-CERT) on incident response activity in Fiscal Year (FY) 2014 reported that out of the 245 total incidents reported across all sectors "roughly 55% involved APT or sophisticated actors"[131].

There are five identified types of attacks, categorized based on the actor's objective(s) as follows: (i) disconnection and goodput reduction, (ii) active eavesdropping (iii) scanning and probing [20], (iv) covert- and side-channel exploitation [130, 132, 133] and (v) code injection [133, 134]. The summary of these attacks types and sub-types are shown in Figure [2.4].

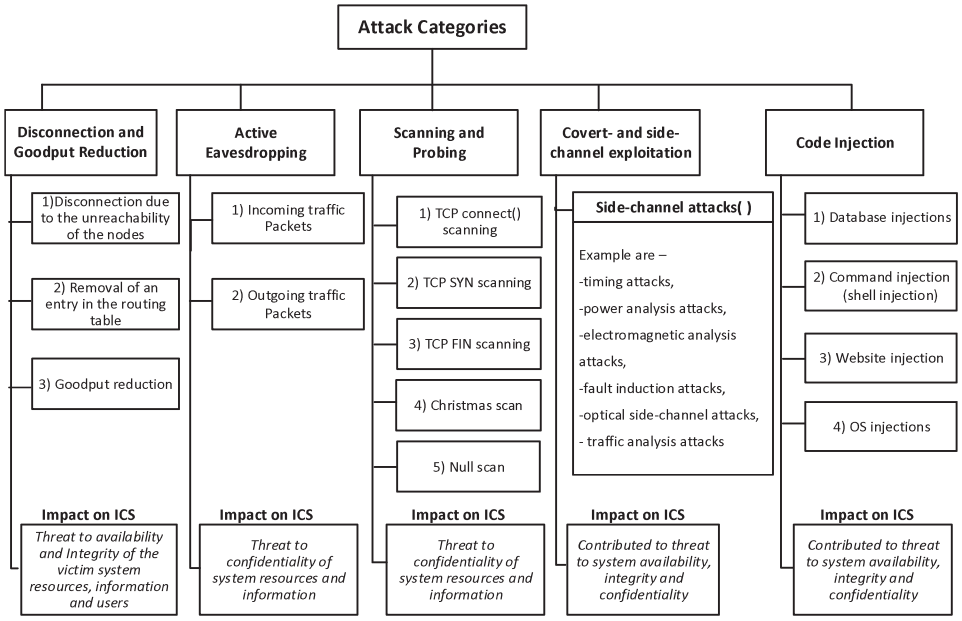


FIGURE 2.4: Classification of cyber stealth attack types and sub-types

- ***Disconnection and Goodput Reduction:*** This type of attack disconnect or isolate a part of the network or reduce its operational goodput. Attacker achieves this by tricking the system parameters into modifying their own behaviour such as modifying their routing tables incorrectly resulting to operational disruption [130]. This type of threat impact on resource availability and sometimes the integrity of the victim system. Disconnection attack can be achieved in many ways:
  - ☐ disconnection due to the unreachability of the nodes,
  - ☐ removal of an entry in the routing table and

- goodput reduction [132].

- ***Active Eavesdropping:*** This type of stealth attack is achieved by the modification of the routing information of incoming and outgoing traffic to hijack traffic in order to eavesdrop on victim's selected nodes. With the gained knowledge through eavesdropping, attackers can modify and compromise the routing tables of nodes on the path between a victim and the sender/receiver casing nodes of the network "disappear" and detouring the network traffic through compromised nodes [130, 132].

- ***Scanning and Probing:*** is a reconnaissance exercise on the network with the aim of port scanning to determine ports that are open, services that may be running on a system and could be available to the attacker [133, 135]. Scanning is useful for both system administrators for security audits and attackers for their bad intention of exploiting and compromising the system network [133].

  Work in [135], classified network traffic scanning into four groups in line with work done by [136].

  - vertical scans,
  - horizontal scans,
  - coordinated or distributed scans and
  - stealth scans.

  Authors, [137, 138] presented a taxonomy of scanning approaches into four groups as follows:

  - statistical,
  - algorithmic,
  - mathematical and
  - heuristic approaches.

  While [139, 140] described different port scanning techniques as follows:

  - stealth scan,
  - fragmentation scan,
  - changes of scan order,
  - slow scan,
  - randomizing inter-probe timing,
  - scan with forged address and
  - distributed scan.

  However, stealth attacks or stealth scans are hard to detect as the objective of the attacker is to successfully perform an attack with a minimal effort [130, 139]. Attackers working from multiple IP addresses attain little IP visibility of their existence and activities, using hard to detect probing techniques to archive this stealth behaviour [139]. Both authors, [130, 139] described different ways system standard behaviour can be modified in order to attain a certain degree of stealthiness: i) TCP connect() scanning, ii) TCP SYN scanning, iii) TCP FIN scanning, iv) Christmas scan, v) Null scan.

- ***Covert- and Side-Channel Exploitation:*** are the physical attacks on a system whereby the attacker exploits the external manifestations of the physical system information leakages, or source of secret data *(side-channel analysis)* such as processing time, power consumption and electromagnetic emission, to identify the internal system computation [130, 134, 141]. This type of attack affects the confidentiality of system information [130].

  Authors in [141], acknowledged existence of two types of software cache-based side-channel attacks as follows:

  - access-driven and
  - time-driven attack.

  In ***access-driven attacks***, the attacker has control over one or multiple spy processes, which share the cache with the victim process. While in ***time-driven attacks***, the attacker may send different encryption and decryption requests to the target crypto process. The attacker records the encryption time upon receiving responses, the variations among the encryption times may provide attackers with enough information to derive the key.

  The authors in [141], also proposed three integrated hardware-software approaches to eliminate source of information leakage exploited in software cache-based attacks as trade-offs between hardware complexity and performance overheads by using:

  - *pre-loading critical data* such as the advanced encryption standard (AES) lookup tables to secure the PL-cache.
  - *informing loads* to protect the RP-cache from time driven attacks.
  - *informing loads assisted software-based random permutation* to replace the random permutation logic in the RP-cache to provide the security protection to regular caches.

  The different possible types of side channel attacks, as explained by [130, 134] are:

  - timing attacks,
  - power analysis attacks,
  - electromagnetic analysis attacks,
  - fault induction attacks,
  - optical side channel attacks and
  - traffic analysis.

  A covert channel is a hidden connection between two cooperating processes that allow them to transmit and receive information in a manner that violates the system's security policy through the channel, without the system noticing. It presents a safe mode for the attacker to send malicious data over the channel or steal any system's information [142]. Example: The TCP/IP covert channel is a type of covert storage channel that uses TCP/IP packet header fields to send hidden data rather instead of a payload field for sending information bits [142]. Authors in [130], highlighted two types of covert channels as follows:

  - communicating extra information to a host and

- hiding the fact that the communication to a host exists.

While [143], classified covert channels into two categories based on the property used for communication:

- *covert storage channels* - uses information slots in packets for transferring data.
- *covert timing channels* - uses relative timing of events for obtaining insight about a system.

- ***Code Injection:*** A code injection-based attack is a type of attack that constitute introducing a malicious or illegitimate code within a computer program, in order to alter system program settings. This makes the operation to change its course of execution, such as a compromise to sensitive data, and executes malicious code without been noticed [130, 144].

  Code injection attacks usually tend to implement some degree of stealthiness, depending on the objective of the attacker, system's characteristics and the channel of transmission. This can be tailored through two main channels:

  - *system vulnerabilities* and
  - *malware infection* (such as malware, virus, or Trojan horses infecting the system.)

    There are different types of code injection, the authors in [130] have categorized code injection attacks, based on the target they are designed to inject, into four groups:

    * database injections also known as *shell injection*,
    * command injection,
    * website injection and
    * OS injections.

  Since the adversary usually aims to retrieve valuable information from the system, or to force a desired malicious behaviour without the end user being alerted, the actual level of stealthiness depends on the objective of the attacker and the way the injection is tailored to the targeted system.

2.3.4 ICS Cyber Security Design Tools

In a quest to find a common framework for an ICS architectural design, development and security implementation, the purdue enterprise reference architecture (PERA) proposed a Purdue model as fundamental guide for designing ICS architecture [145, 146, 147].

Another important resource for cyber security is the MITRE ATT&CK (Adversarial Tactics, Techniques & Common Knowledge) framework, which is a public knowledge base repository of adversary TTPs based on real-world observations [148]. The author in [149], model SCADA attack graph based on the information extracted from ATT&CK using CySecTool editor (CySecTool is a tool used in a given budget for a probabilistic attack graph to finds an optimal cost for security controls portfolio). The extracted information represent multiple attack paths and the probability frequency of the exploited vulnerability. This is to show how security intelligent are stored and used to make security a final decision.

A properly designed intelligent diagnostic system will allow to correct classification of system faults, as stated in [150] that errors made during design contributed to system

failure of approximately 40–45%. These frameworks enable information security professionals and process control engineers responsible for protecting an organization's critical assets to visualize how to protect against a security breach that may involve safety, availability, confidentiality and integrity of critical assets information in use.

2.3.4.1 The Purdue Model for ICS Architecture

The purdue enterprise reference architecture (PERA) was developed from purdue laboratory for applied industrial control at Purdue University by T. J. Williams in 1990's. The Purdue model, shown in Table [2.3], was adopted by ISA-99 PERA as a concept model for ICS network layers architecture as a means of keeping traffic in a control network at deterministic layers as explained in [4, 20, 151, 152, 153]. Purdue Model is a reference model for ICSs network segmentation that defines six layers within these networks, all the components found in the layers, and logical network boundary controls for securing these networks. This model shows the interconnections and inter dependencies of an ICS component for each level. Modified Purdue model was proposed in [4]. However, demonstration of the modified Purdue version are yet to be implemented on securing ICS devices.

TABLE 2.3: Purdue Model for ICS Infrastructure

Level 4	Enterprise/Business Network Layer	Enterprise Security Zone(s)
Level 3.5	Demilitarized Zone (DMZ)	Industrial Demilitarized Zone(s)
Level 3	Site Manufacturing Operations and Control	Industrial Security Zone(s)
Level 2	Area Supervisory Control	Cell/Area Zone
Level 1	Basic Control	
Level 0	Process	

- ***Level 0 (Process):*** Consist of process equipment such as sensors, drives, actuator and robot that are controlled and monitored are configured. The actual process is performed, and product is made in Level 0, hence should not be interrupted to avoid operation failure.

- ***Level 1 (Basic control):*** Is where all the controlling equipment such as DCS, programmable logic controllers (PLC), remote terminal units (RTU), variable frequency drives (VFDs) and dedicated proportional integral derivative (PID) controllers' lives. At this level, input data are received from sensors, processed using control algorithms and sends the output data to a final element. Controllers at this level runs vendor-specific operating systems and are programmed and configured from engineering workstations.

- ***Level 2 (Area supervisory control):*** This Level include the manufacturing operations equipment such as human machine interfaces (HMI), alarms systems and SCADA system such as line control PLC, control room workstations which are targeted toward an individual production area within the system.

- ***Level 3 (Site operations):*** Systems at this level support plant operations and monitoring functions to produce the desired product. The operator interacts with all the production systems through DMZ. Tasks such as quality control checks, managing uptime, monitoring alarms, events, and trends are carried out through HMI system. Also, operational technology (OT) systems reports back up to IT systems at this level. Applications, services, and systems typically found in level 3 includes but

not limited to database servers, application servers, file servers, Microsoft domain controllers and HMI servers engineering workstations.

- ***Level 3.5 (Site business planning and logistics):*** IT systems services that support the production process in facility plant reside at this level. System services reports production uptime and units produced for corporate systems and take orders and business data from the corporate systems to be distributed among the OT or ICS systems. Applications, services, and systems typically found in this level are database servers, application servers, file servers, email clients, supervisor desktops.

  - ***Industrial De-militarized Zone (IDMZ)*** - Act as a communication link between enterprise zone systems and the Industrial zone. IDMZ allows to securely connect networks with different security requirements. It is an information sharing layer among businesses or IT systems in levels 4 and 5, the production or OT systems in levels 3 and lower. IDMZ limits direct communication between OT and IT systems by using broker service in IDMZ to relay communications.

    Systems in the lower layers are not directly exposed to any compromise or attacks. The IDMZ shuts down, if there is any system compromise to reduce the impact on operation and continuity of production. Applications, services, and systems founds in IDMZ are proxy servers, database replication servers, Microsoft domain controllers.

- ***Level 4 (Enterprise network):*** The corporate IT infrastructure systems and applications sits at this level and span across multiple facilities or plants. Data accumulated from the individual plants are used to report on the overall production status, inventory, and demand from this level.

2.3.4.2 MITRE ATT&CK Framework

The MITRE ATT&CK Framework is a public knowledge base repository of adversary TTP based on real-world observations [148]. This knowledge base repository contains wealth of knowledge for development of specific attack detection, prediction and mitigation models. The ATT&CK takes attackers point of view to help organizations or/ cyber security teams to understand the approach taken by attackers to execute successful attacks. Then access the effectiveness of the organizational security operation processes and defense measure in place, as to identify areas that require substantial improvement.

- ***The ATT&CK Model:*** is a set of individual techniques that the attacker can perform to accomplish their objectives, presented by the categories of tactic those techniques falls under. The creation of the first ATT&CK model in September 2013 focused on the Windows enterprise environment. This was released to public in 2015 with 96 techniques grouped into 9 tactics with a complementary knowledge base named Pre-ATT&CK to focus on “exploit” behaviour and ATT&CK for Mobile domain. An increase in ATT&CK growth, recording 219 techniques has lead to extending the focus to Linux and Mac from Windows platform [15].

  As outlined by [154], 440 attack “techniques” belonging to 27 different “tactics” has been produced by MITRE ATT&CK. Each of these techniques contains description explaining what the technique is doing, how it can be “executed”, “used”, and different “procedures” for implementing it. It also contains description on how to mitigate and detect these tactics. This information is essential when developing a defender model to increase the system security.

- ***The ATT&CK Matrix:*** is the visual representation of the relationship between tactics and techniques [155]. Let us take "Execution" for instance from Figure [2.6], as attacker's goal within target environment - there are series of techniques to be implemented in other to deliver this goal. This may include gaining access into the network using spear phishing link. Once the attacker is able to access the system, followed by privilege escalation, credential access, persistence, and finally lateral movement as shown in Figure [2.5]. An attacker has to implement these five techniques as to achieve the main goal [156].

FIGURE 2.5: Five Techniques to Achieve an Attacker's Goal

- ***The MITRE ATT&CK Matrix for ICS:*** This model is an important knowledge base repository that contains the description of actions an adversary may take while exploiting within an ICS network [15]. This knowledge base can be used to better understand, characterize and describe post-compromise adversary behaviour in an ICS.

  All these ATT&CK for ICS does not necessarily apply to all ICS assets due to heterogeneous environments of ICS network containing software and hardware platforms, applications and protocols environments. It serves as an important knowledge base in combination with the Purdue model for ICS to support each individual asset level in terms of cyber security.

- ***The MITRE Enterprise ATT&CK Matrix:*** contains knowledge base tactics and techniques information for the following platforms: Windows, macOS, Linux, AWS, GCP, Azure, Azure AD, Office 365, SaaS [155]. The combination of Pre-ATT&CK and ATT&CK Enterprise tactics ATT&CK roughly align with the intrusion Cyber Kill Chain as visualized in Figure [2.6].

- ***The ATT&CK's Taxonomy Usage:*** MITRE ATT&CK offers a variety of valuable foundational knowledge to cyber defenders and red teamers in a good number of ways that will not only be useful for defensive activities while referencing attackers and their behaviours. It also provides a foundation for penetration testing and gives defenders and red teamers common language platform when referring to adversarial behaviours. Few areas where applying ATT&CK's taxonomy can be useful are as follows:

  * *Mapping defensive controls*
  * *Threat hunting*
  * *Detection & Investigation*
  * *Referencing actors*
  * *Tool integration*
  * *Sharing* and
  * *Red Team/Penetration Test Activities*

  A detailed description of this MITRE ATT&CK's taxonomy usage can be find in [148, 156].

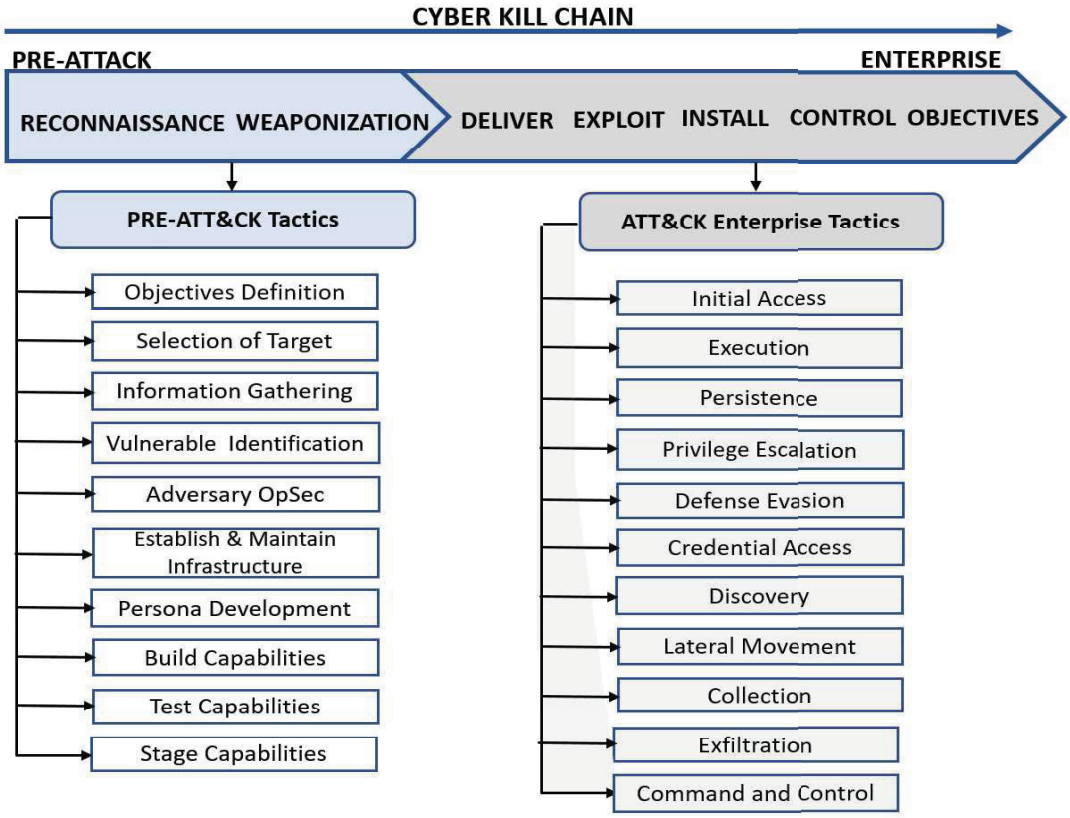


FIGURE 2.6: Pre-ATT&CK and Enterprise ATT&CK

2.4 Intrusion Detection and Prevention Systems (IDPSs)

The IDPSs has taken an active role at the edge of every organisational network. It monitors system events as it occurs and analyses the data for possible indications of attack activities. It detects any attempts made to gain unauthorized access to network system or disrupt system normal operations, and then tries to stop any possible attack incident.

This attempted action made in other to bypass any existing security mechanisms within a system is known as ***intrusion*** [157, 158]. That is to say, any activities that threatens system availability, confidentiality and integrity of that system functionalities is an intrusion. The process of monitoring computers or an organizational network for this intrusion activities, or / modification of files is referred to as ***intrusion detection (ID)***. The action taken to stop or intercept the detected threats is referred to as ***intrusion prevention (IPs)***. The hardware device or software that automates an ID process is referred to as an ***intrusion detection system (IDS)***. The IDSs can respond to malicious events through several ways including;

- ☐ Paging an administrator
- ☐ Displaying an alert
- ☐ Logging the event and so on,

The hardware devices or software that has all the full functionalities of an IDS, and can also attempt to stop an active attack event by responding to the detected threat incidents is termed ***intrusion prevention system (IPS)*** [159, 160]. This can be achieved through:

- ☐ Reconfiguration of security controls, (example: firewall or router) within the systems to block future attacks entry.

- ☐ Reconfigure security and privacy controls in browser settings to prevent future attacks
- ☐ Filtering threatening packets and removal of any malicious events of an attack in a network traffic

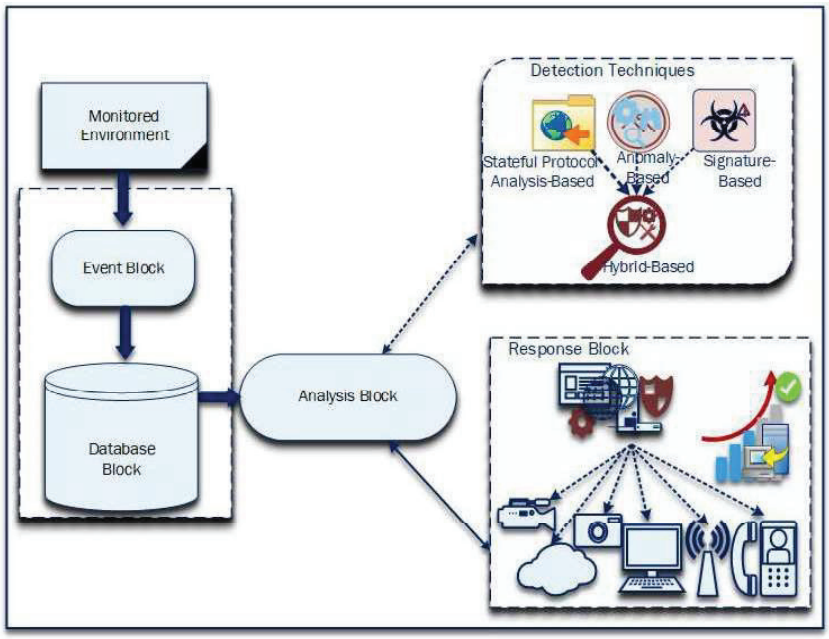


FIGURE 2.7: The Architecture of an IDPS

The general architecture of an IDPSs is shown in Figure [2.7]. It consists of four main blocks: (i) the event block, (ii) database block, (iii) analysis block and (iv) response block. The *event block* is used to gather all the monitored system events. The collected events are sent and stored in the *database block*. The stored events are processed by the *analysis block* based on the deployed detection methodologies. Mostly used detection methodologies are discussed in Sub-section [2.4.2]. If any malicious events are present, alerts will be generated and sent to the *response block*, which will then respond by filtering the threatening packets, remove the malicious events and then reconfigure the security controls so as to block any future attack entry.

2.4.1 Why Deploying IDPSs Technology?

IDPS technologies are deployed to safeguard systems network connections. There are different types of IDPSs technologies as will be discussed in Sub-section [2.4.3], based on the type of events they can apprehend. The typical key functions and uses of IDPSs technologies are as follows:

- ■ **Important Key Function of IDPSs Technologies:**
  - ☐ Record observed events information related to the monitored environment, and then send the observed events data to logging server.
  - ☐ Provides summary/detailed report for a particular suspicious event of interest.
  - ☐ Notify security administrators of any observed malicious events.

- **Important Uses of IDPS Technologies:**
  - Identify any possible attack incident within a system.
  - Able to terminate network connection and block all access to target system, thus stopping any active attack.
  - Monitor file transfers and identify any suspicious file transfer (example: copying a large database onto user's laptop.)
  - May be able to identify and block reconnaissance and notify security administrators, who can then carry out necessary response actions to intercept any possible incidents.
  - May be configured to recognise any security policy breaches.

2.4.2 IDPSs Common Detection Methodologies

IDPSs methodologies are mainly used for the purpose of detection and prevention of any breaches within a monitored system network. The four most widely used IDPSs methodologies are *anomaly-based, signature-based, stateful protocol analysis-based* and *hybrid-based* [157, 158, 159, 161, 162]. These IDPSs methodologies can be implemented in different ways as will be discussed in Sub-section [2.4.3].

2.4.2.1 Anomaly-Based Methodology

Anomaly-based detection methodology is the process of comparing observed activity against a baseline profile, to identify any significant events deviations from what is considered normal behaviour. This baseline profiles which can either be fixed or dynamic can be developed for host, users, system, network connections, applications and so on by observing the behaviour of typical activities while learning the environment over a period. An architectural design of anomaly-based methodology is shown in Figure [2.8].

The fixed baseline profile does not change, while dynamic baseline profile changes if the system events been monitored changes. The IDPSs dynamic baseline profile requires constant update to keep up with the latest system normal behaviour. However, as the system continue to update, it adds more loads to the system, resulting to the system been exposed to evasion. Attackers utilizes this dynamic baseline profile changes and introduces attack little by little over an extended period. The IDPSs unaware of these introduced abnormal events, integrates these malicious events with the baseline profile [157, 163].

- **Anomaly-based: Detection Capabilities**
  - The anomaly-based IDPSs learning stage can be addressed using ML techniques, neural networks, statistical anomaly detection and knowledge/data-mining techniques [161, 162, 164]. Thus, during the detection stage, any deviation from the set threshold (value that sets the limit between normal and abnormal behaviour) is classified as abnormal behaviour. Example: A user accessing a computer about 03:00AM, which is outside business work hour should be flagged as suspicious intrusion.

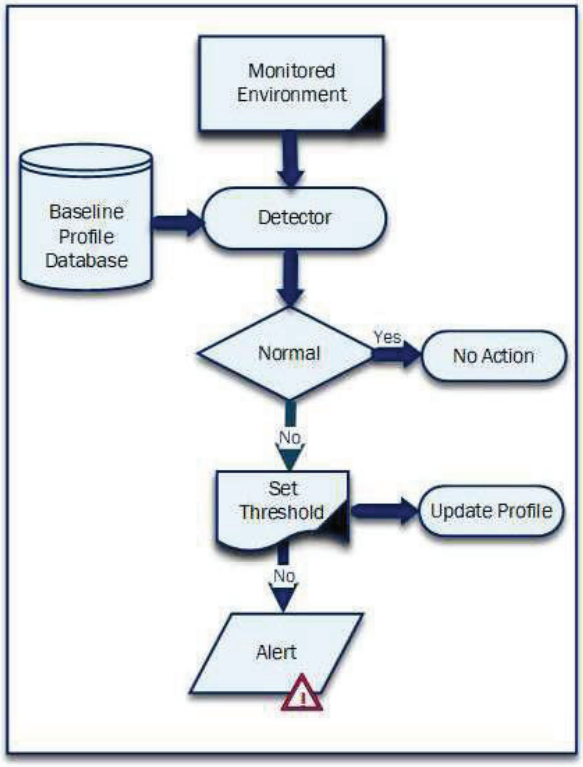


FIGURE 2.8: Architecture of an Anomaly-Based Methodology

■ **Anomaly-based: Advantage and Disadvantages**

□ One of the advantage of anomaly-based IDSPs is that it is very efficient in detecting unknown threats as it does not require prior knowledge of the intrusion. However, the downside is that in addition to possibly generating high false positive rate due to significant deviation of benign events from profiles, raised alert may not contain reasonable explanation of attack incident, making it difficult to validate the accuracy of alert been malignant and not false positive alert.

2.4.2.2 Signature-Based Methodology

The signature-based methodology also known as misuse works by comparing the monitored system events against the list of signatures stored in the signature file/database. These signatures represents pattern of known threats [157, 162]. Signature-base operates by searching the signature file for known signatures and can easily be deployed as it does not require prior knowledge of the environment [165]. If a match is found, the event pattern is reported as an attack, then an alert is raised. Thus, if no match is found, no further action is required as illustrated with Figure [2.9].

■ **Signature-based: Detection Capabilities:**

□ Signature-based detection capability depends on the signature rules [166]. This technique is vulnerable to attack, as the attacker can easily target systems that has not been updated with new signature of modified known attack pattern. Though, signature-base requires significant expert resources to maintain the

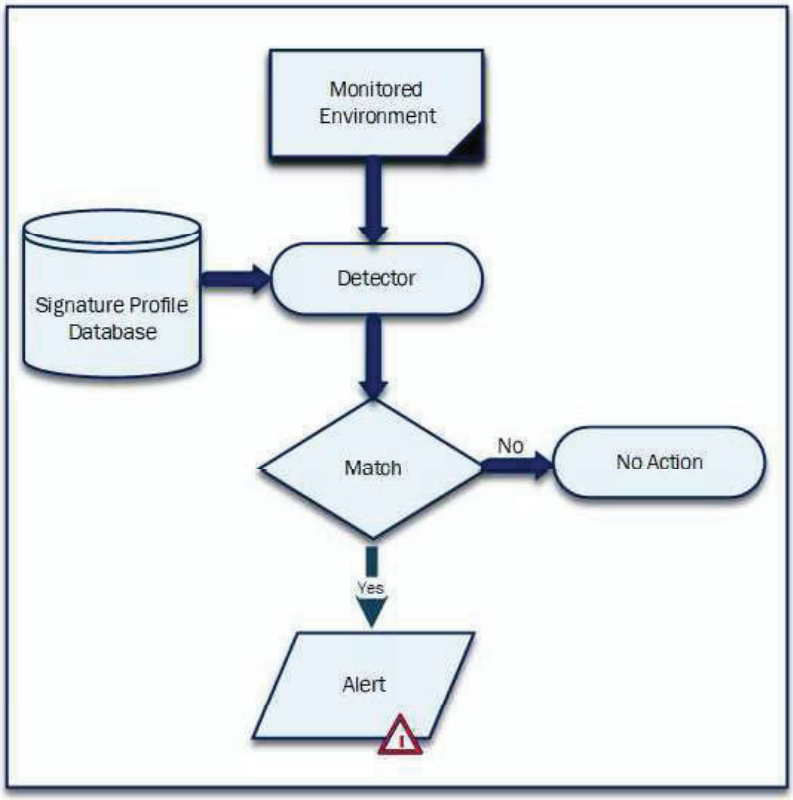


FIGURE 2.9: The Architecture of Signature-Based Methodology

probable endless number of known threat modification to form a new signature, it can be easily maintained by modifying the set signatures/rules to accommodate newly formed signatures [162, 167].

- **Signature-based: Advantages and Disadvantage:**
  - Signature-based detection is very efficient at detecting known threats based on the available list of signatures and it yields low false positives rate[162]. However, it is very ineffective at detecting new unknown threats since the signature database requires constant update to keep up with the latest invented attacks [158]. Example, malware filename modification (from chickdock.exe to chickdock4.exe) by attacker, can mislead the signature-based IDPSs, hence limiting the signature-based from detecting multiple attack, as it will be looking for the original filename “chickdock.exe” [157].

2.4.2.3 Stateful Protocol Analysis Based Methodology

The stateful protocol analysis-based methodology works by comparing the predetermined protocol profile baseline against monitored protocol behaviour. The protocol database contains profile of benign protocol activity representing an accepted protocol behaviour as to identify any protocol deviations [157, 159, 161]. The stateful protocol analysis-based (shown in Figure [2.10]) is built based on protocol standards from software vendors (Example: Internet Engineering Task Force (IETF)). This approach depends on vendors to developed universal profiles which specifies how a particular protocol should and should

not be deployed. Unlike anomaly-based method that uses host or network specific profiles, this enables this approach to have a comprehensive understanding of what a benign protocol should be. However, this method is still open to evasion [157].

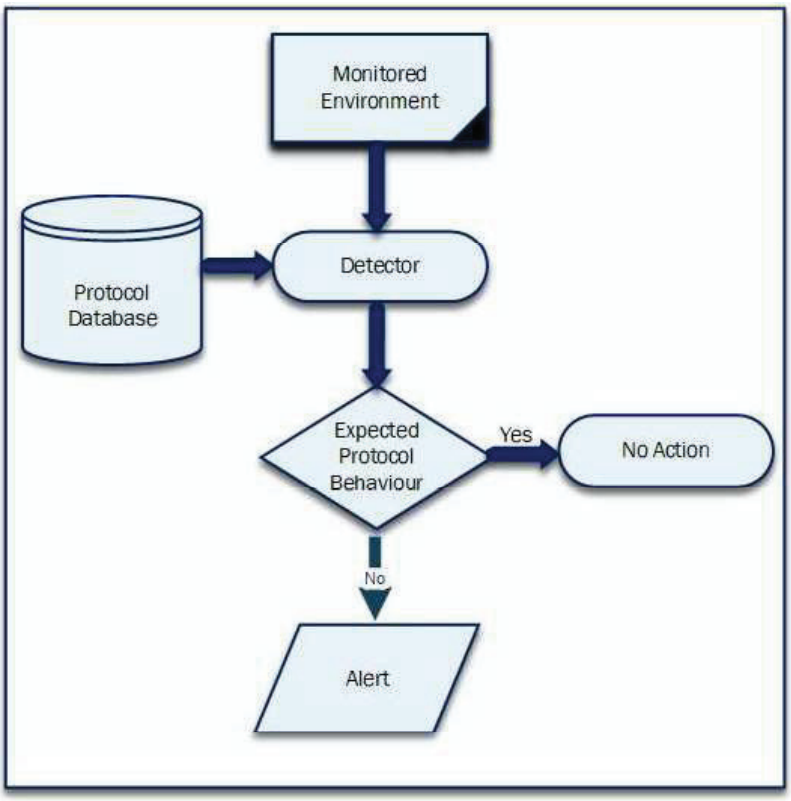


FIGURE 2.10: The Architecture of Stateful Protocol Analysis-Based Methodology

■ **Stateful protocol analysis based: Detection Capabilities**

□ The stateful protocol analysis-based detect attack by monitoring the system protocol behaviour, then compares the observed behaviour with the protocols database, If a match is found, the event protocol behaviour is reported as an attack and alert is raised; however, if expected protocol from the protocol database matches the observed network protocol behaviour, no further action is required [161]. This is illustrated with Figure [2.10].

■ **Stateful protocol analysis based: Advantages and Disadvantage**

□ The main disadvantage of stateful protocol analysis-based methodology is that the existence of other methodologies (anomaly-based, signature-based, and hybrid-based) has reduce the viability of stateful protocol analysis to be used as a standalone IDPS methodology [161]. However, with the indepth understanding of protocol behaviour by the stateful protocol analysis-based, it can perform deep analysis of the interactions between the protocols and applications, this elevates the stateful protocol analysis methodology. It can also identify unexpected sequences of commands; example: constantly issuing the same command over and over [157].

2.4.2.4 Hybrid-Based Methodology

The concept of hybrid-based Methodology is obtained from the combination of other two or more methodologies to provide a better IDPSs performance capability when compared to other individual methods [162, 168, 169]. This method takes advantage of each individual method detection capability to achieve better performance and detects more suspicious events than when individual methodology is used alone [169]. The general architectural design of hybrid-based methodology is represented with Figure [2.11].

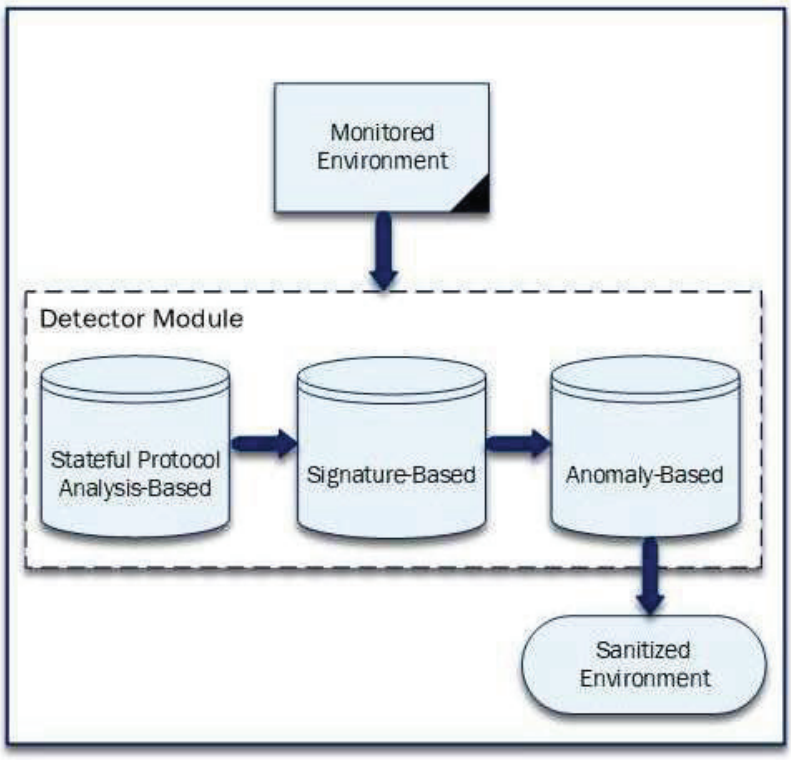


FIGURE 2.11: The Architecture of a Hybrid-Based Methodology

- **Hybrid-based: Detection Capabilities:**
  - As a result of the combination of other methodologies, hybrid-based method utilizes the individual strength of those methodologies to improve its performance output and detection capability. It detects more suspicious events than when a single methodology is used [169].
- **Hybrid-based: Advantage and Disadvantage:**
  - The hybrid-based approach optimizes the visibility and performance of the system. It makes up for the deficiency of each individual methods since it inherited advantages of those methodologies to complement each other's deficiency [161, 169]. However, anomaly-based method (an IDPS methodology which may be used in combination with other methodologies) triggers more alerts than other methods, this makes correlation of alert more complex.

2.4.3 IDPSs Common Technologies

Different types of IDPSs technologies has evolved, these have become an important part of security infrastructure of almost every organization. For the purposes of this thesis,

the following five main IDPSs technologies classified based on the type of events been monitored by them, type of event they can identify, and the ways in which the monitoring techniques are been deployed will be discussed: These are *network-based, wireless-based, network behaviour analysis (NBA)* [157, 158, 159, 170], *host-based*, and distributed-based technologies [162, 166].

2.4.3.1 Network-based IDPS (N-IDPS) Technology

Monitors the network traffic for specific network segments or devices and analyses the network, attributes and pattern of the application protocol activity to identify suspicious activity [157, 159, 162]. It is usually deployed at a boundary between networks, such as virtual private network (VPN) servers, remote access servers, and wireless networks. The N-IDPS typically consist of sensors, one or more management servers, multiple consoles, and may also compose of one or more database servers if N-IDPS supports their usage [157].

- **N-IDPS Security and Detection Capabilities:** The N-IDPS technology possess some useful security features towards the system they support. These security attributes are their ability to:
  - *Gather information* - identifies and gather information about the host, operating system, running application and network activities.
  - *Information logging* - able to log all gathered information related to an important event of interest to be used for further investigation of alert and validation. Logged data is also useful for correlating events between IDPS and other data logging sources.
  - *Threat detection* - utilizes the detection capabilities of the combination of different detection methodologies to produce noticeable detection capabilities. However, the downside of N-IDPS is that considering the events attributes of the monitored environment, it requires a considerable tuning and customization and it has been associated with generating a high FPR and FNR.
  - *Threat prevention* – most IDPS are configured to prevent an attack. However, the N-IDPS network sensor can attempt to terminate an abnormal existing TCP session (process known as session snipping).
  - Provide security information and event management *(SIEM)* capabilities [157].

2.4.3.2 Host-based IDPS (H-IDPS) Technology

Monitors the events and characteristics occurring within a single host to identify any suspicious activity. The monitored events attributes could be system application log (e.g., operating system processes such as startup and shutdown application information), file access and modification, system and application configuration changes, network traffic of that particular single host and so on to identify suspicious activities [159, 162, 166]. This type of IDPS technology is mostly deployed on a critical hosts server that house important and sensitive information[157].

- **H-IDPS Security and Detection Capabilities:**
  - Like every other IDPS, H-IDPS has the capabilities to detect attack, logs information related to the detected attack events. It uses a combination of signature-based detection techniques (to detect known attacks) and anomaly-based detection techniques with set rules (to detect previously unknown attacks).

- It can also detect different types of malicious activity but often causes false positives and false negatives making detection accuracy very hard for H-IDPS [157]. It also requires considerable tuning and customization with constantly updating the H-IDPS policies as the host environment changes.

2.4.3.3 Wireless-based IDPS (W-IDPS) Technology

Wireless-based operates similar to network-based, except that wireless-based monitors wireless network traffic and analyses its wireless networking protocols to establish any protocols that contains suspicious activity. It cannot identify suspicious activity within an application or network protocols such as TCP, UDP etc [157, 159]. This type of IDPS technology is often deployed within organizational wireless network range and sometimes could be deployed within a location that already has unauthorized wireless networking in operation [157].

- **W-IDPS Security and Detection Capabilities:**
  - W-IDPS has the ability to gather information, log events of interest, detect attack, policy violations and system misconfigurations at WLAN protocol level and performs threat prevention [157]. Devices with wireless feature uses computing resources without being physically connected to a network if they are within a certain configured range of wireless network.
  - W-IDPS also enhances user mobility utilizing the wireless technology. To launch an attack, attacker must be within a close physical proximity to the wireless network unlike other type of IDPS that can be executed from any location remotely. Although W-IDPS requires tuning and customization to improve its detection capabilities, the limited proximity of W-IDPS operation, attack detection are generally more accurate. However, the downside is that W-IDPSs are generally unable to resist attack against themselves and cannot detect certain wireless protocol [157].

2.4.3.4 Network Behaviour Analysis (NBA) IDPS Technology

This type of IDPS technology examines network traffic flow to identify malware that generate unusual traffic flows, such as distributed denial of service (DDoS) attacks (e.g., smurf, land, backdoor etc) and policy violations. NBA is usually been deployed to monitor an organization's internal networks flows, organization's networks and external networks such as business partners' networks [157, 159, 166].

NBA technology comprises of sensors and consoles, with some that offers management servers.

- **NBA-IDPS Security and Detection Capabilities:**
  - Like every other IDPS, NBA possesses an extensive information gathering capabilities, logs data related to detected events of interest. Mainly, NBA are based on anomaly-based with some stateful protocol analysis detection techniques for network flows analyses and does not possess any signature-based detection capability. However, administrators can set up custom filters (basically signatures) manually to detect or stop specified threats.
  - NBA technologies develops baselines for expected network flows characteristics from the monitored network environment and constantly updates these

baselines automatically. Thus, tuning or customization are not necessarily required.

- ☐ NBA can accurately detect attacks that generate large amounts of network activity within a short time such as DDoS attacks. However, the detection accuracy varies over time. Although, it is good at detecting significant deviations from normal behaviour as it uses anomaly-based detection methods, attacks can only be detected when the network activity deviation from what is expected is significantly noticeable. This hesitant in attack detection is as a result of inherit nature of anomaly-based methods that is based on deviation from pre-defined baseline [157].

2.4.3.5 Hybrid/Distributed-Based IDPS Technology (HD-IDPS) Technology

Is the type of IDPS technology that combines both network-based and host-based and could exist concurrently [162, 166]. This type of IDPS is also referred to as Distributed-based IDPS [162]. It involves collection of total network traffic information using appropriate agents to monitor a particular computing system. Data analysis is carried out at different locations corresponding to the number of available systems in the network.

- ■ **HD-IDPS Security and Detection Capabilities:**

  - ☐ Each individual IDPS operate totally different from each other. However, integrating multiple IDPS product will increase the benefit of implementing multiple IDPS within organisation minimizing the impact of deficiency of one individual IDPS products.
  - ☐ Implementing integrated multiple IDPS can be performed directly (integrating multiple IDPS from one vendor) or indirectly (integration IDPS using SIEM software). The downside of using SIEM is that logged events data are seen by SIEM after malicious activity may have been displayed on IDPS due to time lag from when event started to when it was transferred to SIEM console impacting on the prompt prevention action [157].

2.5 Deep Neural Network (DNN)

Deep neural networks have emerged as a useful ML tool in Artificial Intelligent (AI) that can mimic human brains data processing function [171]. The DNNs are rapidly developing field of research in different domain, particularly well suited when dealing with very complex supervised and unsupervised tasks which has extended its application to sequence/time-series data [172]. It can automate data feature extraction eliminating required manual human involvement process. Thus, application of ML/DNN-based approach seem to be the current promising methods towards developing an IDPSs technologies in the computational field which can reduce threat and the impact of attacks to an organization in a much needed edge on their sophisticated, less burdened, and more evasive adversaries [173].

The self-learning systems (SLS) have evolved as a result of advancement in mathematics. This SLS has dealt with some of the exiting limitations of traditional approach by utilizing supervisory ML algorithms such as RNN variants that has the potential to identify unknown patterns of suspicious activities within a system by learning the environment behaviour of network traffic events.

These approaches can detect and classify the complex network traffic events directly from data itself more accurately with low false positive rate. RNN variants has demonstrated a notable performance in language modelling, speech recognition [174], symbols recognition in engineering diagram [27], for handling symbols of distinct data types with very little dissimilarity among the symbols and data class imbalance issue, object localization and detection [175], image recognition [176], computer aided diagnosis (CAD) system for myocardial infarction (MI) signals detection using the convolution neural network (CNN) for healthcare system [177], and threat detection [178, 179]. However, the application in cyber security data modelling is still at an early stage of development [180].

Some research efforts have been directed towards generating and processing/labelling large datasets such as NGP[1] dataset [181], the University of New South Wales new benchmark - UNSW-NB15 [2] dataset [182] etc. Nevertheless, in spite of these practical successful application of DNN in different research area, it is important for one to have a good understanding of the mechanisms that drive them for optimal usage, since concrete understanding and interpretation of DNNs is still at the early stage of emerging research field [183].

The idea of neural network application was first mentioned by [184] and the concept of IDPS development in system security measure based on neural network application has been an actively researched area since 1980 [21, 185, 186]. An IDPS based on DNN has the capability to distinguish between multiple attacks scenario from normal events behaviour. Training deeper neural networks is associated with vanishing gradient problem which obstruct the convergence from the beginning since training deeper neural networks are difficult. Authors in [176], has proposed a residual learning framework to mitigate this issue.

DNNs configuration comprises of many parameters of the network such as; hyperparameter selection (e.g, dropout, number of layer in each neuron), network architecture and data processing which helps to establish how well a DNN-based attack detection system will perform. Tuning and customization of these configurations of DNN-based IDPS technologies improve its detection capability. Although, authors in [179] mentioned that optimizing so many parameters simultaneously may lead to overfitting problem, which may result to realizing poor performance accuracy. However, in recent studies, engineering the training set to specifically bias towards the weight across the layer rather than a single layer improves performance output [187].

Recent research has applied DL techniques in cybersecurity [51] for attack detection since it can detect cyber threats by learning the complex underlying structure, hidden sequential relationships and hierarchical feature representations from a huge set of security data. The authors in [188], proposed and evaluated RNN-based model against classical support vector machine classifier (SVM) for cybersecurity in Android malware classification, incident detection, and fraud detection.

RNN emerged as a powerful approach for DL architecture generally applicable for time-series data modelling. Despite the RNN and its variant networks remarkable performance in long standing artificial intelligence (AI) sequence data modelling tasks such as time-series analysis, speech recognition and machine translation [189], applying the same in cyber security task is in early stage of development [180].

[1] NGP: https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets
[2] UNSWNB: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

■ **Let's Define Generic DNN**

The standard DNN is usually composed of an input/output layers and at least one hidden layer within the neural network. Each of this layer is composed of a number of neurons. Each neuron is connected to the neurons in the previous layer and the output of each of this neuron is the sum of the weighted inputs and some bias as expressed in Equation (2.1).

$$a_j^{(L)} = f\left(b_j^{(L)} + \sum_{k=0}^{k} w_{jk}^{(L)} a_j^{(L-1)}\right) \tag{2.1}$$

where L is the current layer of the neural network at the j index into the current layer L, while k is the index into the previous layer $L-1$. The neurons j and k are linked by jk. The neuron j output in layer L, $a_j^{(L)}$ consists of the sum of the inputs from the previous layer $a_j^{(L-1)}$ multiplied by some calculated weights $w_{jk}^{(L)}$, added to a bias $b_j^{(L)}$, and then passed through a non-linear activation function f such as *sigmoid, softmax, tanh or ReLU etc*, as represented in Equation (2.1). The output could be used as an input to the next layer if that layer (currently calculated layer) is not the final or last layer of the neural network.

Given an input sequence $x = \{x_1, x_2, ..., x_T\}$ in the form of a feature vector and its corresponding output data label $y = \{y_1, y_2, ..., y_T\}$, which can be expressed as a *one-hot encoded* vector with each set of vector contains 1 at the index that correspond to the correct data label. Thus, the task of the network is to learn the relationship between this specified set of input/output pairs $f : x \mapsto y$, then adjust the network parameters to minimize the network error. Backpropagation and gradient descent techniques can be introduced to adjust the weights and biases such that the output of the final layer of the neural network lies close to the label with minimal error. The performance index for the network can also be measure to determine how well the model performed.

2.5.1 Recurrent Neural Network (RNN)

Recurrent neural network developed in the 1980s is an effective class of artificial neural network (ANN). It works well on sequence data [52], suitable when dealing with complex supervised and unsupervised tasks [188], ML task where the input and/or output are of variable length [190], sequence-based tasks with long-term dependencies [191] and intrusion detection system [192]. Several studies have indicated that RNN variants has shown promising performance in difficult ML task as highlighted in [193, 194] for a system developed for machine translation.

RNN can be likened to a feed-forward neural network (FFN) [191], with an additional internal feedback loop (short-term memory to store and retrieve previous information over time steps and execute temporal task) which is a circular connections between higher-layer and lower-layer neurons and optional self-feedback connections. These feedback connections enable RNNs to convey data from earlier events to current processing time steps which is in contrasts to FFN. It also uses previous inputs and outputs to adjust the weights W of the networks, thereby creating memory to improve its performance [195, 196].

It maps a variable length sequence of inputs x_t to the outputs o_t, thereby controlling the information flow signal with respect to time step t. Hence, RNN variants are suitable for dynamic real time network events analysis over time steps. Figure [2.12] shows an unfolded architecture of RNN. Given that a network consist of an input layer x, hidden layer s and output layer o, Its mathematical computational flow structure can be represented as in Equations (2.2) - (2.4).

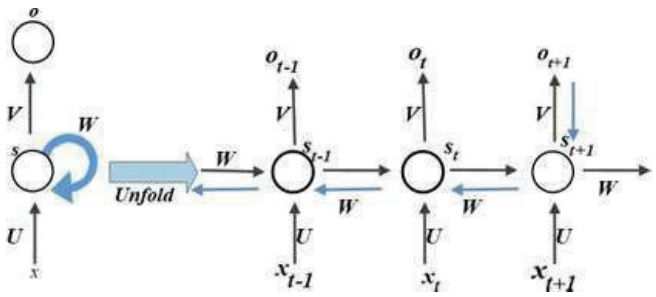


FIGURE 2.12: Schema of Unfolded Basic Recurrent Neural Network [51]

$$s_t = f_\theta(x_t, s_{t-1}) \tag{2.2}$$

$$s_t = f(Ux_t + Ws_{t-1}) \tag{2.3}$$

$$o_t = sf(Vs_t) \tag{2.4}$$

where θ is the tunable parameters that controls information required to be remembered or discarded about the past sequence). f_θ and f are the recursive and non-linear activation functions *(eg. sigmoid, softmax, tanh or ReLU etc)*. The current observed input to the network at time t is denoted as x_t. The hidden state and the previous hidden state at time t is represent as s_t and s_{t-1} respectively. The learnt weight matrices of the input x_t and the previous hidden state s_{t-1} are U_{x_t} and Ws_{t-1} respectively, while the vector of the probabilities of the initial hidden state is set to V in Equation (2.4). The s_{-1} is required to calculate the first hidden state and is usually initialised to zero [197].

Each hidden layer has a non-linear activation function such as *sigmoid* and *tanh* functions as defined in Equations (2.5) and (2.6) respectively. This function is applied to produce the non-linearity value by transforming the input into values that are usable by the final or last layer. These values can also be fed to other stacked recurrent layer.

$$sigmoid\,(x) = \frac{1}{1+e^{-x}} \tag{2.5}$$

where e is the Euler's number.

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1} \tag{2.6}$$

The softmax activation function sf represented in Equation (2.7) is usually applied at the final or output o_t layer as utilized in Equation (2.4) as a non-linearity mapping function from the input features to the output labels. Softmax function ensures that the sum of all values in x sum up to 1. The output can be compared with the one-hot encoded label vector and the corresponding difference between them can also be used to compute the cross-entropy loss L, Equation (2.10).

$$sf = \frac{e^{x_t}}{\sum_{j=1}^{n} e^{x_J}} \tag{2.7}$$

where x_i is the input vector, e^{x_t} is the standard exponential function for input vector, n is the number of classes in the multi-class classifier while e^{x_J} is the standard exponential function for the output vector.

In RNN cyclic connections, each layer represents per time step t information. The unfolded RNN shares weight parameters W across time t (see Figure [2.12]). This indicates the fact that network performs the same task with various inputs over time t. In addition to learning the temporal patterns with cyclic connections, unfolding allows RNN model to learn the association of static features between the input and output sequences. To apply the feedback concept, the back propagation is used to compute the gradients for weight parameters across time steps t.

■ Let's Define Standard RNN

Lets define the standard RNN [198]. Considering the assumption that parameters share the same weight W across the whole sequence in each time step t [199], this reduces the amount of parameters to be trained. Thus, this assumption is utilized to compute the gradient for the weight W parameter across time step t. Lets take $(x_1,...,x_T)$ as the inputs sequence data, each in R^n, $(h_1,...,h_T)$ as the hidden states, each in R^m, and $(y_1,...,y_T)$ as the predicted outputs which may be of variable length, each in R^k.

Hence, at any given time $t = 1$ to T, each one of the hidden state s_t and output layer o_t can be calculated with Equations (2.8) and (2.9) respectively;

$$s_t = tanh(W_{xs}x_t + W_{ss}s_{t-1} + b_s) \tag{2.8}$$

$$o_t = sf(W_{so}s_t + b_0) \tag{2.9}$$

where s_t is the calculated hidden state, *tanh* and softmax *sf* expressed in Equation (2.6) and (2.7) are the non-linear activation functions applied at the hidden state and the output layer in Equations (2.8) and (2.9) respectively. The W_{xs}, W_{ss} and W_{so} are the shared weight matrices, while b_s and b_o are the bias terms for the hidden state s_t and the output o at time step t.

To find the current weights W, the computation of gradient at $t = 2$ will involve back propagating 1 step and add them to find and update the current weight. This technique known as back propagation through time (BPTT) [198], is employed by RNN to reduce network cumulative *error or Loss L*.

The loss as applied on this study is the *cross-entropy loss L*. The loss L function is the sum of all input-output pairs errors $L(x_t, o_t)$ = Cost C_t in a sequence over all the time steps t, with respect to dynamic system network parameters, assume the parameters in each time step are the same. Sharing weights also helps to generalize a model, thus model parameters can be estimated by applying maximum likelihood to minimize the negative log- likelihood of the objective function.

$$L(x,y) = -\sum_t y_t(x)\, log\, o_t(x) \tag{2.10}$$

where y is the true probability distribution, o is the predicted probability distribution, y_t is the actual true probability distribution class value at time step t and o_t is the actual true predicted class distribution value at time step t, x is an input, while log is natural log.

■ Effect of Vanishing Gradients

The generalization[3] objective function is the minimization of the expected future cost C_t and training objective function with less cost C_t effective (negative log likelihood) over

[3]Generalization: The capability of an algorithm to transfer learnt performance from the training set to the test set [190].

observed events sequence at time t. In order to minimize this cumulative network error, RNN uses gradient-based optimization techniques such as BPTT algorithm to compute the gradient.

However, modeling large scale data sequence with RNN and BPTT is not efficient due to vanishing and exploding gradient problem suffered by RNN as introduced in [200]. This vanishing gradients occurs when error is back propagated in many time steps in the deep unrolled RNNs network models, the gradients swiftly becomes very smaller (vanishing gradients) when compared to the current events gradients or explosion of the long-term components, which can exponentially grow (exploding towards infinity) more than short-term.

This problem makes it difficult such that RNN model experiences difficulty in learning relationship between temporally distant events (ie, a specified set of input/output pairs) [201] as a result of ineffective gradient decent on network weight parameters [200]. Other variants of RNN units such as GRU and LSTM [202] have been investigated in the view to finding solution to this shortcomings as "LSTM works well with sequence-based task with long term dependencies" [191]. The forget gate mechanism in LSTM unit helps to alleviate the vanishing gradient when it propagates backward with sequence time series across time as it controls the self-connection unit and regulate the amount of historical information of the memory cell content to be discarded [192].

Let us consider the effect of vanishing gradients on the derivatives of a loss L_t at time t with respect to the parameters as contained in a dynamical system such as RNN with weights W for Equation (2.10). Assuming one want to reduce the negative log-likelihood cost $C_t = -logL(x_t, o_t)$.

The partial derivative of this Loss L_t function as represented in Equation (2.10) with respect to network parameters using chain rule and by applying BBTT to estimate the gradient can be access from Appendix [B].

2.5.2 Long Short-Term Memory (LSTM)

LSTM unit introduced by [202] is a second order RNN designed with augmented recurrent gates known as forget gates (FG). It was motivated by error flow analysis in RNN to deal with RNN vanishing gradient problem [203]. LSTM has the capability to handle sequence-based tasks with long-term dependencies [191, 204] such as but not limited to robotic control [205], speech recognition [206, 207], machine translation [193] and handwriting recognition [204].

LSTM is composed of a set of recurrent connected subnets referred to as a memory block. It uses this memory to generate complex, realistic sequences containing long-range structure. This memory block is a complex processing unit that is composed of one or more memory cell and three multiplicative gates known as *input, output and forget gate* with purpose in-built memory cells (recurrent connection value 1 as *constant error carousel (CEC))*. This value will be active across the time step t and is triggered when a memory block has not received any value from outside signals [190, 208].

To combat the issue of vanishing gradient that prevents RNN from learning long term dependencies through gating mechanism, these multiplicative gates controls the information flow to and from the unit over long periods of time. As long as the gating units are shut, gradients can flow through the memory unit without any modification for an indefinite amount of time thereby avoiding the exploding and the vanishing gradient problem [209].

The formation of an LSTM network is the same as that of simple RNN with exception that the hidden layer non-linear units are replaced with memory blocks in LSTM network. Let's define the LSTM model with forget gates. The computation of recurrent hidden state s_t can be seen as mean elementwise multiplication as illustrated in Equation (2.11).

Given that there are hidden units s, the batch size is n, and the number of inputs is m. Thus, the input at time step t, is $x_t \in \mathbb{R}^{n\times m}$ and the hidden state at previous time step $_{t-1}$ is $s_{t-1} \in \mathbb{R}^{n\times s}$. Hence, the input i, forget f and output o gates at time step t can be defined as follows: the input gate is $i_t \in \mathbb{R}^{n\times s}$, the forget gate is $f_t \in \mathbb{R}^{n\times s}$, and the output gate is $o_t \in \mathbb{R}^{n\times s}$. These can be computed at each j^{th} index into the LSTM layer time step as follows:

$$\begin{aligned} i_t^j &= \sigma(x_t W^{xi} + s_{t-1} W^{si} + b_i)^j, \\ f_t^j &= \sigma(x_t W^{xf} + s_{t-1} W^{sf} + b_f)^j, \\ o_t^j &= \sigma(x_t W^{xo} + s_{t-1} W^{so} + b_o)^j, \\ \tilde{g}_t^j &= \tanh(x_t W^{xg} + s_{t-1} W^{sg} + b_g)^j, \\ c_t^j &= f_t^j \odot c_{t-1}^j + i_t^j \odot \tilde{g}_t^j, \\ s_t^j &= \tanh(c_t^j) \odot o_t^j \end{aligned} \tag{2.11}$$

where σ and tanh are used as the activation functions. The $W^{xi}, W^{xf}, W^{xo}, W^{xg} \in \mathbb{R}^{m\times s}$ & $W^{si}, W^{sf}, W^{so}, W^{sg} \in \mathbb{R}^{s\times s}$ are the weight parameters, while $b_i, b_f, b_o, b_g \in \mathbb{R}^{1\times s}$ are the bias parameters. The $\odot$ is the pointwise multiplication, while c_{t-1} and c_{t-1} are the memory units at the current unit and previous unit, respectively. The s_t and s_{t-1} are the hidden layer at current time t and output value of each memory cell in the hidden layer at previous time $t-1$ respectively. The elements of LSTM cells as shown in Equation (2.11) are discussed as follows;

- **The input gate** i_t^j - accept activation from the current data point x_t and the value of the hidden layer at the previous time step s_{t-1}. It's value is used to multiply the value of another node. The input gate regulate the extent to which the new memory content is passed through to the memory cell. Thus, if the gate value is zero, flow from other node is terminated whereas, if the gate value is one, all flow is allowed to pass through.

- **The forget gate** f_t^j - regulated the extent to which the existing memory is forgotten. It is the mechanism used to reset the content of the memory cell, (ie., the ability to decide when to remember and when to ignore inputs in the hidden state).

- **The output gate** o_t^j - regulated the amount of memory content to be exposed. It's value can be calculated by the value of the internal state multiplied by the value of the output gate, then passed through activation function to produce the output of each cell.

- **The input node** $\tilde{g}_t^j$ - accept activation data from input layer x_t at current time t and also from hidden layer at the previous time step s_{t-1}. It uses a tanh function with value range of $(-1, 1)$.

- **The memory cell** c_t^j - also known as *internal state* has a self-connected recurrent edge often known as the *CEC* that spans adjacent time steps with constant weight, allowing error flow across time steps without vanishing or exploding. It operate with

two mechanism; the input gate to regulate the amount of new data to be considered through g_t^j and f_t^j to regulate how much of old memory content $c_{t-1} \in \mathbb{R}^{n \times s}$ to keep.

- **The hidden state s_t^j -** $s_t \in \mathbb{R}^{n \times s}$ is the output of the LSTM unit, (ie., the gated version of the tanh (activation) of the cell). The values of s_t is usually within the interval of $(-1, 1)$. If the value of the output is approximately 1, all cell information are passed through to the predictor, otherwise if it is close to (0), information within the cell are retained with no further action [209].

The LSTM uses these gates to determine whether to keep the existing memory cell. Intuitively, if an important feature is detected from an input sequence, the LSTM has the capability to carry this information over a long-term, thus, capturing potential long-term dependencies [191, 209].

2.5.3 Gated Recurrent Unit (GRU)

GRU proposed by [197] is gating mechanism in RNN that is comparable to LSTM. It shares with the LSTM unit the ability to better model and capture long-term dependencies and it's performance on polyphonic music modelling and speech signal modelling are like that of LSTM with fewer parameters [210].

A GRU unit has two gating unit; a reset gate r and an update gate z units that regulate the flow of information inside the unit from the previous activation to newly computed candidate activation.

- **The update gate z_t^j** - determines how much of the unit updates it's content, or activation. The GRU exposes the whole state each time as it does not have any mechanism to regulate the extent to which it's state is exposed [191].
- **The reset gate r_t^j** - if the reset gate is off (ie., close to 0), it can causes the unit to behave as if it is reading the first data of an input sequence, thereby allowing the unit to forget the previously computed state. The computation of r_t^j is similar to that of update gate.
- **The candidate activation $\tilde{s}_t^j$** - computation of $\tilde{s}_t^j$ is similar to that of the traditional recurrent unit.

The GRU unit update gate z_t^j and reset r_t^j at time step t at the j^{th} index level can be computed as expressed in Equation (2.12).

$$\begin{aligned} z_t^j &= \sigma(x_t W^z + s_{t-1} W^z)^j \\ r_t^j &= \sigma(x_t W^r + s_{t-1} W^r)^j \\ \tilde{s}_t^j &= \tanh(x_t W^s + (s_{t-1} \odot r_t) W^s)^j \\ s_t^j &= (1 - z_t^j) s_{t-1}^j + z_t^j \tilde{s}_t^j) \end{aligned} \tag{2.12}$$

where σ and tanh are the activation functions and $\odot$ is an element-wise multiplication.

2.5.4 Decision Tree Classifier (DTC)

Data mining is a useful process for extracting interesting information from dataset by applying different types of ML classification techniques such as decision tree classifier (DTC), support vector machine (SVM) and naïve Bayes. Threats detection model based

on ML techniques has been used to detect and classify unknown attack, and to differentiate attacks into attack family [211].

Classification process is a data mining method that involves categorizing a few data instances into group based on the old data trained on where class labels are known. Different classification algorithms have evolved based on their mathematical techniques such as DTC and neural networks for analyzing data in several ways to make its prediction. Classification process are of two type: Lazy (e.g. naïve Bayes) are those type of classifier that do not build any model prior to testing. Aactive (e.g. decision trees) are those type that are can create a model from training data [212].

The DTC is one of the data mining classification techniques for ML-based model commonly used. This method takes time to build the tree by dividing the classification problem into sub-problems. It builds a decision tree which in turn is used to develop a model for classification or prediction purposes. It comprises of root node, inner nodes and leaves nodes. The inner node performs a specific task to decide which branch to follow to the next inner node until a leaf node is reached and the final decision is made. It is non-parametric algorithm that is capable of dealing with large complex datasets effectively without imposing any complicated parametric structure [213].

The DTC approach has been applied in many areas such as but not limited to developing classification and prediction model [213], and sleep staging [214]. Authors in [214] proposed a framework for sleep staging that consist of a multi-class SVM classification based on decision tree approach. This study realised "mean specificity, sensitivity and overall accuracy of 0.92, 0.74 and 0.88 respectively compared to visual scoring".

Chapter 3

Detection and Defence of APTs Attack Approaches

***

3.1 APT Detection Methods 47
3.2 Existing Proposed APT Detection Systems 49
3.3 Existing APT Detector's Limitations 56
3.4 Summary of the Reviewed Detection Methods 59
3.5 Experimental Datasets 61
3.6 Preliminary Experiments 69

In this chapter, a few existing approaches tailored toward attacks and APT detection were discussed under three sub-headings: (i) APT attack detection and prevention, (ii) defending APT attack and (iii) APT attack mitigation. This chapter also contains an overview of few limitations with regards to the discussed approaches. The ATT&CK Matrix TTPs and some helpful outgoing network traffic attributes that can be utilized to uncover malicious activities were also highlighted. The four different domain datasets used for the four case studies of this thesis, it's attack types and representations as aligned to the study were also discussed. In addition, preliminary experiments carried out over the course of this study are presented.

3.1 APT Detection Methods

In general, the main goal of an attack detection model is to detect and distinguish malicious activities from normal system activities. A system can be compromised, and devices within the compromised system network can become infected. Diverse IDPSs options can be used to respond to these malicious compromise through several ways as discussed in Section [2.4].

There are numerous attack vectors and TTPs being utilized by APT attackers to achieve their goal. Each TTP represent one possible method to realize a particular step of APT campaign as illustrated in Figure [2.6]. For example, persistence capability of APT in a compromised Linux system requires eleven well defined TTPs to achieve this adversary objective. Each TTP represents a possible sequence of lower level actions in the MITRE ATT&CK framework highlighted in [148]. Examples are Pre-OS Boot, modification of boot scripts, and installation of a rootkit etc. Pre-OS Boot technique consist of three

sub-techniques, used to describe how persistence is achieved before an operating system boots, these are Bootkit, Component Firmware, and System Firmware, documented in The ATT&CK for Enterprise Matrix [155].

3.1.1 The ATT&CK Matrix Techniques Tactics Procedures (TTPs) Concept

The techniques, tactics and procedures concept of ATT&CK Matrix are:

- ***Techniques:*** is an important component of TTP concept that represent "how" an adversary perform an action to achieve their tactical objective. Example, dumping credentials from an operating system to gain access to useful credentials within a target network. This type of adversary operation can be broken down into sub-techniques of specific behaviours that describe how those behaviours can be executed to achieve tactical objectives. Examples, *OS Credential Dumping* - consist of accessing LSASS Memory, the Security Account Manager, or accessing /etc/passwd and /etc/shadow distinct behaviours [148], [155].

- ***Tactic:*** is an important component of the TTP concept that represent the "reason" for an ATT&CK technique or sub-technique, i.e. the adversary's tactical objective for performing an action. It contains contextual categories and serves as a guide for what techniques should be during operation such as persistence, data discovery, delivery, laterally movement, file execution, and data exfiltration. Example, *execution* - is a tactic that represents sub-technique in execution of adversary-controlled code on remote or local system [148], [155].

- ***Procedure:*** Another TTP concept is "procedure", this represent the specific implementation adversaries have used for technique or sub-technique, i.e. how adversaries uses techniques and sub-techniques to achieve their objectives. Example, the use of *PowerShell* to inject into lsass.exe to dump credentials by scraping LSASS memory on a target system such as *OS Credential Dumping* [148], [155]. Other examples of procedures are Process Injection and LSASS Memory which are all well defined behaviours.

3.1.2 Useful Outgoing Network Traffic Attributes to Uncover Malicious Activities

Detection of network security breaches can be achieved through various methods such as IDS and SIEM (Security Information and Event Management) tools. Among APTs objectives are to access and extract sensitive and valuable data from their target organization; analyzing organizations' in/out flow of network data may help to track down C&C attack messages. Some of the outgoing network traffic attributes that can be utilized to uncover malicious activities are:

- ***Data Transmission period:*** Presence of malicious activities can be discovered if a particular machine persistently and continuously generate a certain type of network traffic over a given period of time through a pattern that does not correlate with normal network traffic [110].

- ***Network Connection duration:*** APTs attack activities involve generation and extraction of large events data achieved over a long established connection within the targeted organization network [215]. Thus, as normal network connections are rarely established for longer periods of time, it is very essential to check and know the duration of normal organization business activities network connections, when monitoring network connection duration.

- ***Amount of Outgoing Data:*** Understanding organization's common network data flow in a normal business circumstance is very important for monitoring outgoing data. Normal system users typically send small amounts of data when they establish connections and receive large amounts of data, while infected machines may generate larger amounts of unusual outgoing data. It will be beneficial to consider organizations normal in/out flow data in a normal business circumstance when analyzing outbound data [216].
- ***Country Destination IP addresses:*** If analyzing outgoing IP packets and its destination addresses are in a country that have no association with the organization, this could be an indication that the connections may have been established by malware.
- ***Frequent communication with new Domains:*** Repeated communication with an unknown new domain could indicate malware presence. APTs attack usually registers numerous new domain names to aid with the creation of redundant copies of C&C channels [110], then uses dynamic DNS to route all traffic between these multiple infected systems. This makes it more difficult to track and trace such traffic.
- ***DNS cache:*** When establishing network connections, users usually use URLs by either clicking on a hyperlink or typing the URL, the users' DNS resolver will find a matching IP address for the URL, this IP address is then stored in DNS cache for a certain period of time. Any outgoing connection without a corresponding IP address entry in the DNS cache, may indicate malware infection [110].
- ***Malicious sites:*** Keeping a blacklist of sites that have hosted malicious content will help to track devices that are still in communication with such sites, this may indicate presence of an attack. However, this will not prevent communication with any new established malware servers. However, communication with such server will be more prone to be detected.

3.2 Existing Proposed APT Detection Systems

The ability of an intrusion detection system to detect every possibility of an active attack on a system is a security issue. There have been a number of successful breaches of CI. Few examples of APTs attack as discussed in Sub-section [2.1.5] are: - ***Stuxnet*** which is an APT attack aimed against uranium enrichment facilities in Iran [69, 217], ***Deep panda*** [98, 99] and ***Epic Turla*** [100, 101].

Different techniques and frameworks have been proposed and successfully implemented to address these security issues [1]. These proposed approaches have led to a significant pool of solutions geared towards addressing security and systems resilience. One of this detection model is intrusion kill chain (IKC) model, created by Lockheed Martin analysts in 2011 to support a better detection and responding to attacker's intrusions. This model can serve as a great building foundation and a concept to start with [218].

However, it is very improbable to stop cyber breaches through prevention approach alone. Discovering an unusual behaviour within a system that may represent security breach in real time will provide an opportunity for the target organization to respond to such a situation. According to [219], there exist different approaches to deal with specific threat, however, none of this approach is good enough to safeguard a system as there is always new evolving threats to deal with at any point in time. Hence, the author highlighted the important of developing " a multi-faceted, joined up approach" that possess breach detection capabilities, since security breach can only be managed effectively if discovered on time.

3.2.1 APT Attack Detection and Prevention

Threat detection in a specific CIs was carried out by [220] using a hybrid of two neural network learning algorithms – the Error-Back Propagation and Levenberg-Marquardt, for normal behavior modeling to develop an IDS using Neural Network based Modeling (IDS-NNM). This model was achieved by developing window based feature extraction technique; construction of training data using randomly generated intrusion vectors from real network data. Their result shows the ability of IDS-NNM to detect long and short intrusion attempts consisting of several packets and achieved good detection rate while generating no false positives when evaluated with previously unseen data.

A framework for ranking internal hosts as to identify and rank suspicious hosts that could be involved in data exfiltration activities related to APTs. By deploying a prototype of this framework, the author realized ranked list of suspicious hosts that could be involved in data exfiltrations and other APT related activities [221].

Authors [222] proposed DTB-IDS: an IDS based on decision tree using behavior analysis that can minimize APT attacks damage by executing quick detection of APT attacks. The authors evaluated their system using data collected from API features of malicious code behaviour and achieved accuracy detection rate of 84.7%.

Study in [83], proposed an approach "HetGLM", a GNN-based intelligence for detecting LM attack within a system. The LM is one of APT attack steps. The authors constructed a heterogeneous graph with different network entities to detect any malicious LM through discovering anomaly links by considering rich semantics in heterogeneous graphs. The GNN-based anomaly link detection algorithm is designed to spot lateral movements within a given network. This approach uses metapath-based sampling procedure and attention mechanism to analyse information from several network traffics as to detect anomaly links. The author reported that this approach was able to detect LM attack using public dataset. However, the validity of this model performance in real time scenario is yet to be confirmed.

The author in [223], proposed an adaptive scanning technique, called DNS-cache based scanning. A scan detection system (SDS), that monitors the incoming and outgoing flows of enterprise network subnet to detect scanning probes based on the correlation of flows with preceding DNS query and responses. This approach was of the view that without reducing the time to live (TTL) values of resource records in DNS responses, attackers can piggyback on cached DNS records to bypass detection; thus, TTL reduction mechanisms were integrated in their approach to improve the effectiveness of the authors' approach by reducing the expiration time of cached DNS records, against stealthy and adaptive scanners. Although the SDS focus on detection of single attack scanning probes, it recorded a 96% accuracy using public dataset of network flows.

An approach named SPuNge was proposed in [224, 225]. The architecture of this system aims to detect potential targeted attacks within a network for further investigation by processing threat information collected from actual users such as consumer and server machines. This approach uses a combination of clustering and correlation techniques to identify groups of machines that share a similar behaviour with respect to the malicious resources they requested, such as exploit kits, drive-by downloads, or C&C servers and by correlating the location of infected machines and industry in which they operate. Examples are Oil & Gas companies and Government etc. SPuNge was evaluated with one week's threat data collected by Trend Micro from over 20 million users installations worldwide. An overview of SPuNge system is shown in Figure [3.1].

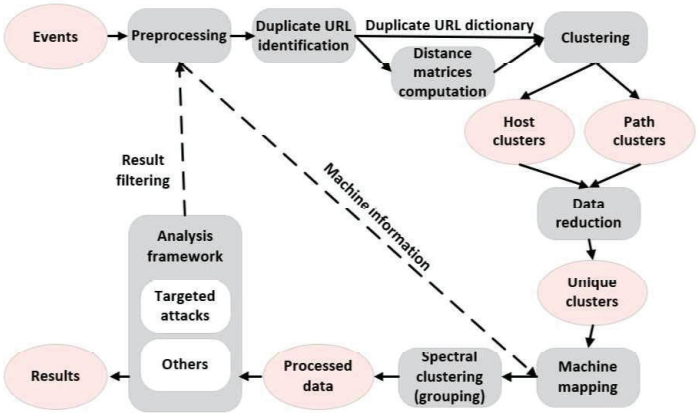


FIGURE 3.1: SPuNge Architecture

Again, as a result of new insight of C&C derived from analysis of C&C features, [41] proposed new features of C&C tagged independent access feature into DNS records, to characterize the communications difference between C&C and normal HTTP requests. This study assumed that C&C domains access tends to be independent, while accesses are correlated in case of legal web domains access, which makes it possible to distinguish C&C domains from legal domains. They achieved average detection rate of 89.30% of C&C domains connection against 79.00% obtained in [226] while applying the feature of periodic connection only. Both works focus on detecting the C&C domain stage of APT attack using the same dataset of APT infection discovery challenge sponsored by Los Alamos National Laboratory (LANL) [227].

Also, an approach based on semi-supervised learning were proposed in [74]. This approach considered a complex networks characteristic of APT activities as evolving APT-Attack Network (APT-AN). Enterprise network data consisting of ***17,684*** hosts from the LANL were used to determine the performance of this method by deploying a state machine to model domain node changes over time. Their focus was to identify any changes or any infected hosts within the network during the APT attack process and achieved an average precision rate of 90.50% for the three APT stages.

A multi-stage attacks framework approach based on combination of the IKC attack model [77] and a hypothesis based approach of known patterns was presented in [228] to support the detection and analysis of multi-stage cyber-attacks. This framework as shown in Figure [3.2] comprises of three main components: (i) a multi-stage attack model, (ii) a layered security architecture and (iii) a security event collection and analysis system. For the purpose of data collection and handling big data, this framework was further divided into 5 modules (logging module, log management module, malware analysis module, intelligence module and control module) using Hadoop. Log records of ***69,969,233*** fed into Hadoop distributed file system (HDFS) and 5 Hadoop nodes were used. The framework was tested with simulated attack data. It took 7 minutes and 38 seconds processing time to get the information based on the IDS alert. This framework looks very promising as it took care of different stages of APTs lifecycle. However, testing it with real time data installation to validate the ability of any model developed based on this framework in real time APT scenario and improving the human interface contribution as to facilitate experimentation with different correlation algorithms is lacking.

An approach based on ML alert correlation framework (MLAPT) system was introduced by [229]. MLAPT approach focuses on detecting different techniques used within each stage of the APT lifecycle. It is based on three phases: (i) Phase 1: *threat detection*,

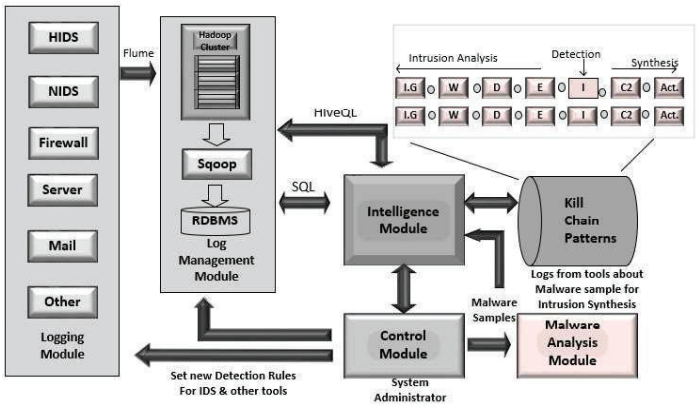


FIGURE 3.2: Multi-Stage Attack Framework Overview

eight modules have been developed to detect various attack techniques used at different APT steps. (ii) Phase 2: *alert correlation*, in which a correlation framework has been designed to correlate the alerts produced in phase 1, links the outputs of the detection methods to a single APT scenario. (iii) Phase 3: *attack prediction*, based on application of ML prediction module on the alert correlation output. This approach is believed to be capable of processing and analysing network traffic in real time and has been evaluated on network traffic data and pcap files containing attacks attaining 84.80% prediction accuracy. Nevertheless, the validity of MLAPT framework on real APTs is subject to further investigation. It's prediction module relies on data correlated over a six months period or more which depend on the number of correlated generated cluster. It also has high false positive rate (FPR) and not able to detect all the steps of APT lifecycle.

With the belief of mapping APT lifecyle phases into cyber kill chain, [230] proposed Markov Multi-Phase Transferable Belief Model (MM-TBM) for predicting data exfiltration phase of APTs. Author in [231], utilised the Hidden Markov Model (HMM) statistical model for a given sequence of correlated alerts to estimate the most likely sequence stages in APT lifecycle, to predict the next step of the APT campaign. However, Markov process is based on systems behaviour, which is the future state X_{n-1} of a discrete time $n+1$, is dependent on the current state X_n only and not on the previous states $(1,2,\ldots,n-2,n-1)$ [230]. This work addresses the multi-stage attacks (MSAs) campaign. It involves two phases and is an extension of work in [229] that dealt with phase one - attack scenario reconstruction correlation based on matching the attributes of the elementary alerts generated over a specified time window. While in the second phase, the decoding module (DM) applies the Baum-Welch algorithm on the correlated data to train the HMM transmission, emission and for generation of initial probabilities. It then uses the Viterbi algorithm to determine the most likelihood of the sequence of APT stages, and finally apply the forward (FW) algorithm to predict the next stage of APT. Although this work achieved a commendable estimate of the sequence of APT stages with accuracy prediction of at least 91.80%, with 43.60% accuracy of prediction based on two observations. This low percentage result derived from next step prediction, with small samples of APT_alerts_dataset [232] used, made it clear how difficult it is to detect different stages of APTs. This is a known global challenge that has attracted lots of research interest.

Active Learning Base Framework (ALBF) for malicious .pdf detection work in [38], focused on detecting malicious executable .pdf file at the point of initial penetration. The authors used a total of ***50,908*** PDF files, containing ***45,763*** malicious and ***5145*** benign files, from four different sources. These sources are (i) VirusTotal repository contains ***17,596*** malicious files, (ii) Srndic and Laskov [233] contains ***27,757*** malicious files, (iii)

Contagio project contains 410 malicious files and Ben-Gurion University contains 0 malicious file and 5145 benign files. SVM is employed as a classification algorithm with the radial basis function (RBF) kernel in a supervised learning approach. It achieved TPR of 99.80% in differentiating malicious files from benign files. Although this study achieves a good TPR, it detects only one step of APT lifecycle.

Authors in [234], proposed disguised .exe file detection (DeFD) module, which aims at detecting disguised .exe files that are been transferred over the network connections. DeFD detection is based on a comparison between the multipurpose internet mail extensions (MIME) type of the transferred file and the file name extension. This technique was evaluated using SkypeSetup.exe and ViberSetup.exe which were disguised into SkypeSetup.pdf and ViberSetup.doc for DeFD to detect.

3.2.2 Defending APT Attack

By applying a game theory concept, the defender can predict future threats [113]. Author in [114] took this approach to propose a two-layer differential game framework. This approach takes into consideration the joint threats from APT attacker and the insiders to characterize the interaction, among attackers, insiders and defender as two layer differential game in an open-loop settings. They identified response strategies for each player by optimizing their long-term objectives and demonstrated the existence of Nash Equilibrium (NE) for each game through analysis. This framework could be considered as useful tool for practical system security design facing APT and insider threats. Also, [115], proposed a hybrid strategy game-based on dynamic defense model to optimally allocate constrained secure resources against APTs in a wireless sensor network. They proposed a data fusion method “NetF” which is used to verify the existence of NE in the game, then device a response strategies for different nodes at different times, as to detect the possibility of a node being a malicious node.

Authors [116] developed an approach based on optimal control theory in view to address APT defence problem. To formulate this approach, the authors modeled APT problem as an optimal control problem, formulating an APT defence strategy as a control, representing the state of evolution of an organization as a set of dynamic constraints, quantify the effectiveness of an APT defence strategy as an optimal control problem. The authors also concluded that the "expected loss and overall cost of an APT defence strategy should be appropriately balanced to adapt to specific application scenarios". Although this strategy can be applied to time-varying networks, there is no demonstration of this approach application either on a simulated APT attack or real time attack scenario.

Considering the impact on APT data availability for active decision and limited generalization for new threats constrain that many available solution relies on, [235] introduced domain adaptation APT malware log samples to target domain by fusing and leveraging various data sources to combat APTs using APT driven Bayes Net (APT-BN) techniques. This approach enables the use of system-based features that seek an indication of compromise within the APT-EXE[1] data from anomaly perspectives or activity threshold.

An assumption that if any stage of APT lifecycle fails, the entire plot to execute an APT attack will also fail, thus, [236], propose an approach based on monitoring all accesses to unknown domains for detecting APT attacks. This approach was motivated by the hypothesis that users are been lured into downloading malware by redirecting users to fake domains with the intent for malware to spread once download is made, hence detecting such unknown domains comes with high accuracy.

[1] APT-EXE: https://github.com/aptresearch/datasets

Also Brogi and Tong in [42] believed that detecting an APT involves keeping track of different steps taken over the period and linking the relationship between the elementary attacks triggered during APT lifecycle. This was tackled by retracing these links through observing the information flow tracking (IFT) using TerminAPTor detector which put emphasis on the links between the traces left by attackers in the monitored system during the different attack stages. This study demonstrates the chain of attacks that exist within this monitored targeted system. However TerminAPTor depends on an agent to detect those elementary attacks, this agent could be a standard IDS.

Generic unsupervised batch and streaming anomaly detection algorithms in system provenance recorded traces to detect APT-like activities was proposed by [112]. This work was validated on provenance traces gathered from four different operating systems. However, anomalous analysis, interpretation and identification processes are based on manual process. Also, not all anomalous events are APT like attack activities, hence requires suitable threshhold to limit effort.

3.2.3 APT Attack Mitigation

When systems are under attack, time become of great value at the face of this adverse organised cyber threats. Mitigation of threats is constrained by numerous elements such as use of obsolete manual change processes and patchwork infrastructures that comprised specialist point solutions, lack of resources and skilled security professionals in the field [107], the strength and deterministic nature of the APTs attacker, attack duration [52], [11] & [55], internal employees & environment or infrastructure in use [55].

To overcome and mitigate threats within a system, author [107] suggested that an organisation should increase the maturity level of the orgnizaitional security within that system by implementing an automated response system as the workhorse of the rapid response process. This will be orchestrated in a manner that automation plays the greater role in providing a solution that will exploit “machine-speed responses and allow either autonomous automation or human mediated automation, to deploy workflows and pre-approved change processes to achieve an immediate, foreseeable effect in protecting information security programmes. The author believed that an automated response across systems can instantly hunt down automated mass attacks across systems, thereby minimizing the number of successful attacks.

Study in [108] took a different strategy in finding a better strategy towards mitigating APT in a network by identifying some vulnerabilities of APT attacks using multiple simulated attacks in a virtualized environment. Their findings indicated that updating the antivirus software and the operating system with the latest patches may help in mitigating APTs, but the strongest defence system could still be infiltrated by APT attack vectors. The authors arrived at this conclusion from an experiment carried out by simulating three types of APT attack vectors (namely: malicious USB attacks, backdoor attacks and drive by download spear phishing attacks). Three types of exploits were selected per APT attack vector, making a total of nine APT attack scenarios. Anti-virus, sandbox, firewall and browser setting approaches were used as defensive mechanism. Five cases were identified where a specific defence mechanism has failed to detect and prevent the APT attack. However, this experiment is implemented in a virtualized environment and lacks the use of zero-day malware. Also the simulated attack vectors were limited to three APT threat vectors, tested with four countermeasures.

While [109], proposed APT mitigation process using multi-level security access control (MLS-AC), motivated by security risk associated with Bring Your Own Device (BYOD)

concept. The author suggested that user's data access restriction should be implemented based on user that poses behaviours that can initiate APT attach such as; (i) downloading software & or file and storing it on work drive, (ii) accesses company's information while on public network, (iii) prone to creating or deleting .pdf and .doc files classified as high risk files, and (iv) uses work device to access social network or malicious application.

APT attack follows an organised planned steps or stages as discussed in Sub-section [2.1.3]. Each individual stages can be detected through several probabilities. [110, 111, 112] suggested that system security can be improved by educating and creating awareness among system users & system administrators as well as web-based communities[2] users on different attack vectors thereby equipping them with necessary information and knowledge, thus contributing towards system protection. This could lead to reduction in the number of successful attack, depending on users' level of awareness about actions that can introduce a threat to an organization.

3.2.4 Deep Learning Detection Approach

A method based on DL based using Bidirectional Long Short-Term Memory (BiLSTM) and Graph Convolutional Networks (GCN) was proposed by [237] to detect APT attacks. This technique is employed by analysing network flow into IP-based network flows, and then reconstruct the IP information before features extraction of APT attack detection of compromised IP.

Authors [101] proposed a deep learning classification approach for APT attack classification and detection. In their work, they have implemented "a 6-layer deep learning model with 4 hidden layers sized of 50 x 50 in the 10 epoch range by 10-fold cross validation method" on the NSL-KDD dataset. Their study focused on classifying two classes where normal events are grouped under normal class and all the attacks group found within NSL-KDD dataset are grouped as anomaly and achieved 98.85% accuracy with FPR of 1.13.

MalNet detection approach was proposed in [39]. This approach automatically learns features from the raw data to capture the malicious file structure patterns and code sequence patterns using DNNs for malware detection. This approach used a generated grayscale image data, extracted from raw binary file. Convolution neural network (CNN) is used to learn from grayscale images, the structure features of a malware from its local image patterns and opcode (operation code) sequence is extracted by decompilation tool, LSTM network is used to learn features about malicious code sequences and patterns. Once CNN and LSTM have completed the features learning process, the MalNet then generates a binary classification result for malware detection by applying stacking ensemble which joins the two networks' discriminant result with an extra metadata feature. The authors evaluated MalNet and achieved detection accuracy of 99.88% and TPR of 99.14% with a false FPR of 0.1%. Figure [3.3] is an overview representation of Malnet architecture.

The authors in [238], considered a case of nonlinearities in communication data flow in an Automatic Generation Control (AGC) system. Applied stacked RNN-LSTM model as a detector and classifier in order to detect False Data Injection (FDI) attacks in AGC systems and achieved accuracy of 94%. The authors focused on three types of attack; the ramp, step and pulse attacks since attacks in AGC target frequency deviation signals and tie-line power signals. Accuracy, sensitivity, specificity and precision were calculated

[2]Web-based Communities: is an important place where people seek information and share expertise. It has attracted millions of users, many of whom have integrated these activities into their daily practices [111].

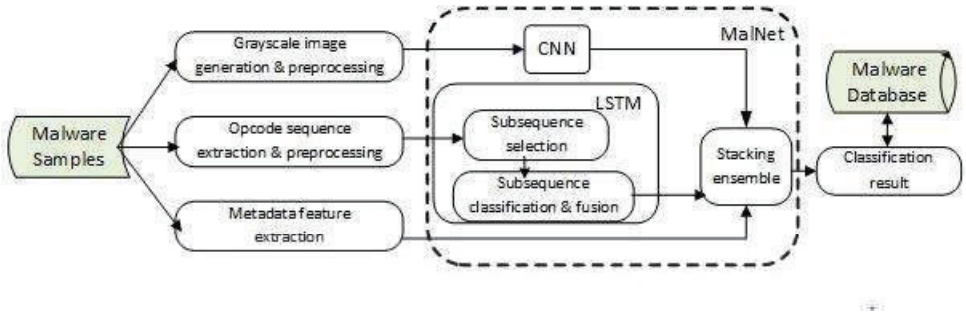


FIGURE 3.3: MalNet Architecture

as metrics measure in order to validate detection and classification capability of this approach. Two case scenarios were implemented using RMSprop optimizer to enhance the model performance and realised a precision of 99.01% and 99.22% indicating that this approach as a classifier, performed very well.

3.3 Existing APT Detector's Limitations

There are a number of difficulties and challenges that make APT attack detection approaches highly inefficient. Few of these challenges as encountered over the cause of this thesis include; lack of standard available public datasets containing real APT attacks scenarios, information sharing, multi-class problem, data imbalance, class overlap, high classification error and performance comparison, etc.

However, APT also exhibit several limiting characteristics that make it extremely more difficult to detect than any other traditional threats. To overcome some of these challenges, recent studies and proposals often tends to generate their own experimental datasets as highlighted in Table [3.1], while others also apply data re-sampling techniques on (either simulated or existing) data for dealing with imbalance data caused by unevenly distributed events elements among classes [239]. Based on this thesis, few of these limiting encountered factors are as follows:

- ***Absence of publicly available & accessible data:*** A comprehensive dataset that contains real normal and abnormal behaviours are required to carry out a proper evaluation of network IDPS effectiveness and performance [240]. Due to the specific dynamic nature of APT attack, that does not follow a unique pattern, this makes network security even more critical when APT scenarios are been considered. Majority of the reviewed approaches used different datasets as can be seen in the Table 3.1. Availability and accessibility of dataset containing realistic APT scenario have become even a more challenging issue when testing and comparing APT detection models [241].

- ***Information sharing:*** Most companies, businesses and organization who are victim of APTs attack find it difficult to share information which may be helpful to cyber security professional due to the fear of them passing information logs and details of such an attack events to an unknown adversary unknowingly.

- ***Multi-class problem:*** such as detecting multi-steps behaviour of APTs is a difficult challenge unlike when dealing with two class classification. APT attack is executed through low and slow approach using combination of different sophisticated techniques that span over long periods of time with very rare events that relates to APTs

[1]. As a result of these rare events, FPR tends to be extremely very high [221], making it difficult to detect rare attack. Detecting the rare attack is a classification problem with data imbalance [1].

- ***High classification error:*** Considering high FPR resulting from difficulty in classifying rare attack as presented in multi-class problem, work in [1] demonstrated that uneven data distribution is not the only factor that affects performance of many proposed techniques for attack class classification. In their work, the achieved overall average detection rate for minority class of interest *U2R* in KDDCup99 dataset improved from 63.20%, 68.40% & 57.60% to 78.90%, 78.90% & 73.70%, while in the case of detecting *Worms* in UNSW-NB15 dataset, implementing synthetic minority oversampling technique (SMOTE) approach did make an impact as the overall average detection rate improved from 0.03%, 0.06% & 0.09% to 32.50%, 35.50% & 23.2%.

  Nevertheless, the detection rate obtained are still below average, hence, suggesting that a high classification error also contributed to the poor performance of the validation matrices used to evaluate classifiers' performance [1, 242].

- ***Imbalanced data:*** Due to the amount of huge network traffic information flow, APTs utilize the unevenly distributed nature of this huge data elements to hide within weak signals among huge traffic data resulting to data imbalance. This makes it difficult to detect rare APT attack. As mentioned by [221], only few hosts are infected during an APTs attack, suggesting that C&C APT attacks type are more harder to detect.

- ***Class overlap:*** Overlapping of different classes as highlighted by Vuttipittayamongkol et al. in [243], also have a higher impact on the detection and classification of imbalance datasets due to the low visibility of the rare class attack tactics than the dominance of the majority classes. This makes it difficult to detect those rare attack step, demonstrating that class overlap has a negative impact on the model leanring process, thus affecting the performance of the implemented detection model.

- ***Performance Comparison:*** Majority of these frameworks and approaches discussed achieved a significant result in line with their respective domain of focus. However detecting only one step of APT lifecycle does not infer a complete protection of CI and information system. Considering the complexity of CI such as ICSs system, there exist some constraints as summarized below:

  - ☐ Most of the approaches reviewed used different dataset as outlined in Table [3.1], were brief description of each individual model, limitation(s), dataset used and reported outcomes / goals are highlighted. This makes it difficult to compare and validate model performance.
  - ☐ Significant expert knowledge is required for setting up and maintenance.
  - ☐ Can be easily evaded when the infected hosts connect to the C&C domains while users are surfing the Internet.
  - ☐ Depends on an agent to detect elementary attack and has high FPR.
  - ☐ Adversary also has access to the publicly available MITRE ATT&CK knowledge base.
  - ☐ Adversary also could use the Purdue Model for ICS to understand indepth knowledge of ICS architecture so as to fortify their knowledge.

3.3.1 Summary of Reviewed Proposed APT Detection Systems

An overview of the discussed existing proposed approaches and frameworks for APT detection, prediction, response and mitigation are summarized in Table [3.1].

TABLE 3.1: Summary of Reviewed APT Detectors

APT Detection Approach	Description	Dataset Used	Limitation	Reported Detection Rate (DR) / Goal)
APT driven Bayes net (APT-BN)	Enables use of system-based features that seek an indication of compromise within the APT-EXE by adapting APT malware log samples to their target domain [235].	APT-EXE dataset	Limited to .exe file with target domain and has not been tested in real time for APTs scenario	Reported TPR of 0.80%
Active Learning Base	Detects malicious PDFs based on whitelists and their compatibility as viable PDF files [38]	Four different data sources (VirusTotal repository, Srndic and Laskov [233] Contagio project and Internet & Ben-Gurion University (randomly selected)	Detects only one step of APT lifecycle	Reported TPR of 99.80%
CODD - C&C approach based on application of the independent access feature into DNS records	Application of the independent access feature into DNS records, to characterize the communications difference between C&C and normal HTTP requests. It considers the access to the C&C domain independent while the access to the legal domain is correlated [41]	APT infection discovery challenge - LANL [227]	Investigated one stage of APT technigues; C&C It can be easily evaded when the infected hosts connect to the C&C domains while users are surfing the Internet	Reported 89.30% average accuracy of C&C
Contex-Based Detected Framework	Models APT as a pyramid in which the top of the pyramids represent the attack goal, and the lateral planes indicates the environments involved in the APT lifecycle [40]	Collected data from several security feeds over an hour to build users profile	Significant expert knowledge is required for setting up and maintenance	Model APT as an attack pyramid based on different events per planes path
Combined DL (MLP, CNN & LSTM) approach	Network traffic analysis based on combined deep learning model to detect APT attack signal by extracting IP features from network traffic flow & classifying APT attack IP [244]	Collected data from 29 network traffic files in the Malware Capture CTU-13 dataset	Detection is limited to extracted IP features from information flow based on two class classification (either; normal or IP attack)	Reported 93.00% to 98.00% accuracy achieved
Data Leakage Prevention	Detection of data exfiltration step using DLP algorithm [245].	Used data not reported	Detects only one step of APT lifecycle and cannot deliver in real time detection	DR of 94.00%
DL classification & detection approach	A 6-layer DL model with 4 hidden layers in 10 epoch range by 10-fold cross validation method for two classes [101]	NLS-KDD dataset	Focused on two class classification (normal and abnormal classes)	Reported 98.85% accuracy & FPR of 1.13
Framework based on ranking internal hosts for possible of activity related to in APT attacks	Framework for ranking internal hosts for likelihood of been involved in APT attacks activity by monitoring network traffic for suspicious APT-related activities specific to data exfiltrations [221]	Used data not reported	Focused on outgoing traffic generated by internal hosts. Detect only few hosts that show suspicious activities but not actually the infected hosts	List of ranked most suspicious internal hosts
MLAPT	Uses event correlation analysis that run through 3 phases: a) threat detection, b) alert correlation and c) attack prediction [229]	APT_alerts_dataset.db	High False Positive rate, not able to detect all the steps of APT lifecycle	Prediction accuracy of 84.80 %, - described APT stages based on lifecycle
Multi-Stage Attacks Framework	A Multi-Stage Attacks framework based on combination of the IKC attack model [77] and a hypothesis based on known patterns [228]	Simulated attack data	This framework has not been tested in real time APT attack scenario	Reported 91.80% accuracy

Table 3.1 Continued from previous page - Summary of Reviewed APT Detectors

APT Detection Approach	Description	Dataset Used	Limitation	Reported Detection Rate (DR) / Goal)
PADASYN algorithm (improve version of Adaptive Synthetic (ADASYN) Sampling	Proposed PADASYN algorithm for data balancing is an improve version of ADASYN data sampling approach [239].	Generated three different datasets comprises of white traffic generated by normal software, while malicious traffic is generated by multiple APT samples in an isolated sandboxes environment	Focused on the two new features added to their own generated dataset for PADASYN evaluation	Reported F1-score of 0.98 & 0.94 for the two dataset used respectively
Statistical APT Detector	Correlation of generated events in each APT step in a statistical way [246]	LANL dataset	Requires significant expert knowledge to set up and maintain	Observation suggests that -monitoring any single part of the attack chain is not sufficient to identify attack - multiple behavioural indicators of an intrusion will be attributable to a single host
Spear-Phishing	Focus on "Tokens" and utilises mathematical and computational analysis to filter spam emails [247]	Sample data that contains spam email	Detects only one step of APT lifecycle and the Spear Phishing email may not contain any of the Tokens	87.00% accuracy with Email classification error of 13.00% was reported
SPuNge	Event data collection and analysis from the hosts' side utilising combination of clustering and correlation techniques to identify groups of machines that share a similar behaviour with respect to the malicious resources they access [224]	Data feed - collected users' data from an antivirus vendor (information on the malicious URLs users access over HTTP or HTTPS.)	Detection of one step of APT lifecycle only and cannot deliver in real time detection	Timing measurements indicates that SPuNge can handle large amount of data.
TerminAPTor	Observes information flow tracking (IFT) to find links between the elementary attack trigger [42]	Used data not reported	Depends on an agent to detect elementary attack and has high false positive rate	This study claims that every single attack can be detected by a traditional intrusion detection system, hence TerminAPTor was created out of the intuition that observing information flows between traces left by the attacker during every single step (attack).

3.4 Summary of the Reviewed Detection Methods

So many frameworks and approaches have been proposed and tested towards finding a better way to detect, respond and mitigate the impact of cyber attack, especially that of APTs. Most of these approaches as reviewed focuses on detection of attack with respect to a specific domain which can be easily evaded when the infected hosts connect to the C&C domains [41]. Some detects only one step of APT lifecycle [38], while others requires a significant expert knowledge for setting up and maintenance purposes [40]. Others depends on an agent to detect elementary attack and has high false positive rate [42], detects single technique used during APT attack such as HetGLM [83] and SDS [223]. This technique could also be a benign action such as Tor communication network connection used during APT attack campaign for data exfiltration, can as well be used legally to protect the privacy and confidentiality of user traffic [90, 248, 249, 250]. Some of these approaches

are based on ML techniques [51], DL approach [237], hybrid of different methods, while others considered specific features such as application of independent access feature into DNS records [41], monitoring all accesses to unknown domains [236], and implementation of MLS-AC [109].

Author [239] proposed PADASYN algorithm for data balancing, which is an improved version of Adaptive Synthetic (ADASYN) data sampling approach. The author believes that addition of new features to APT organization will improve the accuracy rate of APT detection. In their experiment, they combined two new features exhibiting similar features to classify APT attack traffic by using 10 DNS features, 11 TCP and HTTP/HTTPS features to construct a feature set with AdaBoost algorithm. Also, [1] proposed combination of SMOTE data oversampling techniques, together with heterogeneous ensemble approach based on DL for rare APTs attack detection.

APT attack detection based on a multi-layer analysis [251], markov multi-phase transferable belief model (MM-TBM) [230], ranking internal hosts as to identify and rank suspicious hosts [221], machine that share similar behaviour [225], network information flow and network traffic analysis based on combined deep learning model [244] , IP host, while others focus on specific attack step such as C&C [41, 226].

Although, most sophisticated tools employed by APT attackers enables them to completely evade observation, modify their behaviour so as to avoid been detected and maintain their presence thereby minimizing anomaly detection scores [112]. Nevertheless, most of these approaches have realized commendable results that could serve as a starting point for building a strong, reliable and resilient system to safeguard CIs. Hence, combination of these individual techniques will portray a clear APT campaign. Correlating individual output of these detection techniques applied in each step to build a framework to combat APT attack, could result to reduced false positive rate of the developed detection system and will be a great research area.

Considering that APTs campaign operate with multiple steps to achieve their goal using different TTPs at each step, this suggest the need to implement a multi-step approach that will consider different APT steps as most of the existing approaches either deals with specific threat or detect one step of APT lifecycle only. Bearing this in mind, detection of a single APT step, does not suggest detection of APT campaign, adversaries uses several TTPs to achieve their goal. This study suggest that detecting an APT involves keeping track of different steps taken over the period and linking the relationship between the individual attack alert triggered during APT lifecycle. Hence, the identified research area to develop and implement a multi-step APT attack detection approach to mitigate against APT attacks.

This thesis developed a novel multi-step APT attack detection framework, termed APT_{DASAC} based on DL.This framework has been deployed in four different domains to investigate APT attack steps detection capability of this system. APT_{DASAC} goes through three different stages, these are; data input and probing stage, analysis stage, and decision stage. This approach takes into account, the different TTPs used at each step to complete an APT scenario, and developed algorithms that are able to process the network traffic, identify, classify, and correlate each step to attack group. This differentiate this work from other state of the art methods and is a novel approach in terms of dealing with APT attacks detection as a multi-attack activities and as oppose to single attack activity and using ensemble deep leaning model to correlate different outputs and detect APT attack steps in my opinion.

3.5 Experimental Datasets

This Subsection discussed the datasets used for this study. Due to the specific dynamic nature of APT attack, that does not follow a unique pattern, availability and accessibility of dataset containing realistic APT scenario have became a challenging issue when testing and comparing APT detection models. For this thesis, **APT_alerts_dataset.db**[3], the **NGP**[4], **UNSW-NB15**[5] and **KDDCup'99**[6] datasets will be used to evaluate the proposed APTs detection framework in line with APTs lifecycle as described by Eke et al. in [5]. These datasets are available for research purposes.

3.5.1 APT_alerts_dataset.db

The APT alert dataset contains APT attack traffic, simulated by triggers from different malicious activity observed over the APT life cycle. It contains **3676** alerts records with eleven different types of alerts and eight detectable alerts of which 403 alert records represent an incomplete alert as result of correlation time window has elapsed before the current correlated alerts completes an APT campaign (see Figure [3.4]. Each alert consist of seven attributes: *alert_type* - alert type, *timestamp* - alert time, *src_ip* - source IP address, *dest_ip* - destination IP address, *src_port* - source port, *dest_port* - destination port and *infected_host* - infected host IP (see sample log in Figure [3.5]).

The eight detectable alert as contained in this simulated data are *disguised_exe_alert, hash_alert, domain_alert, ip_alert, ssl_alert, domain_flux_alert, scan_alert, tor_alert*. These alerts are grouped into four detectable steps of APT lifecycle as highlighted on this section. These eight detectable alerts aligns to different adversary TTPs of the MITRE ATT&CK model discussed in Sub-section [2.3.4.2], in combination with the IKC attack model explained by authors in [77] and [148].

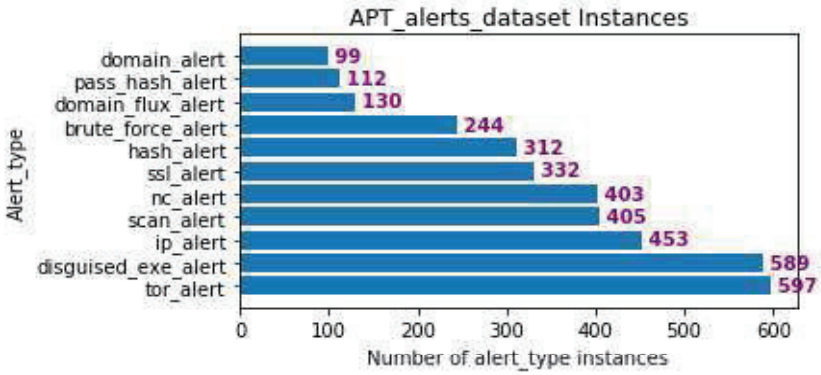


FIGURE 3.4: APT_alerts_dataset Records Distribution

- **The Four Detectable Alert Steps Group:**
  - S1_alert ∈ {disguised_exe_alert, hash_alert, domain_alert}
  - S2_alert ∈ {ip_alert, ssl_alert, domain_flux_alert}
  - S3_alert ∈ {scan_alert}
  - S4_alert ∈ {tor_alert}

[3]APT_alert: https://repository.lboro.ac.uk/articles/dataset/Dataset_of_Advanced_Persistent_Threat_APT_alerts/7577750
[4]NGP: https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets
[5]UNSWNB: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/
[6]KDDCup'99: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

alert_id	alert_type	timestamp	src_ip	src_port	dest_ip	dest_port	infected_host
3329	domain_flux_alert	1463014480	149.170.187.32	57676.0	149.170.39.93	53.0	149.170.187.32
451	tor_alert	1460369993	149.170.88.12	63263.0	149.172.108.111	443.0	149.170.88.12
2414	ssl_alert	1456455966	149.170.228.161	57224.0	18.216.228.214	443.0	149.170.228.161
962	ip_alert	1455308418	149.170.18.83	54921.0	202.44.54.4	443.0	149.170.18.83
2101	disguised_exe_alert	1453176914	149.170.130.77	56246.0	150.47.193.156	80.0	149.170.130.77

FIGURE 3.5: APT_alerts Log Sample with Attributes

3.5.2 The New Gas Pipeline (NGP) Dataset

The NGP data is generated through network transactions between a RTU and a MTU within a SCADA-based gas pipeline at Mississippi State University. This data was collected by simulating real attacks and operator activity on a gas pipeline using a novel framework for attack simulation as described in [53]. The data contains three separate main categories of features: the network information, payload information, and labels.

The *network topologies* and the *payload information* values of SCADA systems are very important to understand the SCADA system performance and detecting if the system is in an out-of-bounds or critical state [7]

3.5.2.1 Three Main Features of NGP Dataset

- ***Network Information*** - This category provides a communication pattern for an IDS to train against. In SCADA systems, network topologies are fixed with repetitive and regular transactions between the nodes. This static behaviour favours IDS in anomalous activities detection.
- ***Payload Information*** - This provides an important information about the gas pipeline's state, settings, and parameters, which helps to understand the system performance and detecting if the system is in a critical or out-of-bounds state.
- ***Labels*** – is attached to each line in data to indicate if the transaction within the system activity is normal or malicious activities.

3.5.2.2 Cyber Attacks as contained in the NGP data Records

The NGP data contains ***214,580*** Modbus network packets with ***60,048*** packets associated with cyber attacks. Each record contains 17 features in each network packet. These attacks are placed into 7 different attack categories containing 35 different specific type of attacks. These attacks categories are represented in Figure [3.6] and Table [3.2].

These 7 attacks categories are further grouped into 4 attacks categories (see: Figure [3.7)] to align with APT steps.

- ***Response injection attacks*** - contains two types of attacks, naïve malicious response injection (NMRI) (which occur when the malicious attacker do not have sufficient information about the physical system process) and complex malicious response injection (CMRI) (these type of attack is designed to mimic certain normal behaviours using physical process information making it more difficult to detect).

[7] Modbus: https://simplymodbus.ca/TCP.htm Accessed on 10/03/2021

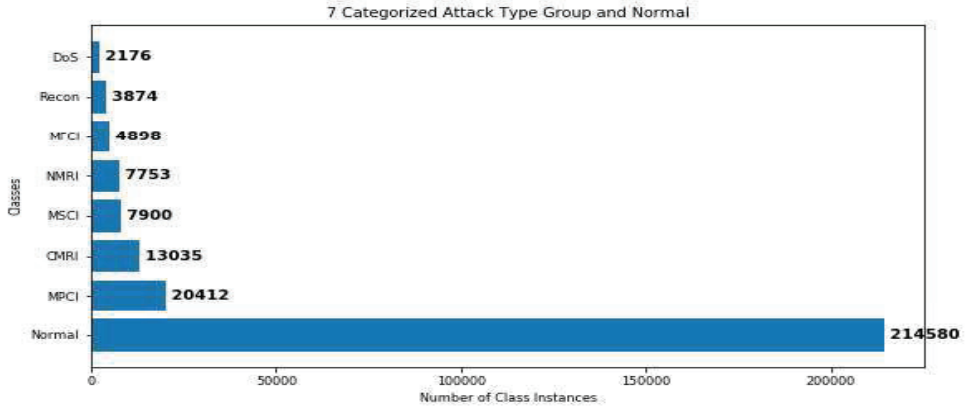


FIGURE 3.6: NGP Data Records Distribution

TABLE 3.2: Attack Categories with Normal Records Type

Attack Categories	Abbreviation	Values	APTs Step
Normal	Normal	0	Not Applicable
Naïve Malicious Response Injection	NMRI	1	Delivery
Complex Malicious Response Injection	CMRI	2	Exploitation, Exfiltration
Malicious State Command Injection	MSCI	3	Data Collection, Exploitation
Malicious Parameter Command Injection	MPCI	4	Data Collection, Exploitation
Malicious Function Code Injection	MFCI	5	Data Collection, Exploitation, Exfiltration
Denial of Service	Dos	6	Data Collection, Exploitation, Exfiltration
Reconnaissance	Recon	7	Reconnaissance

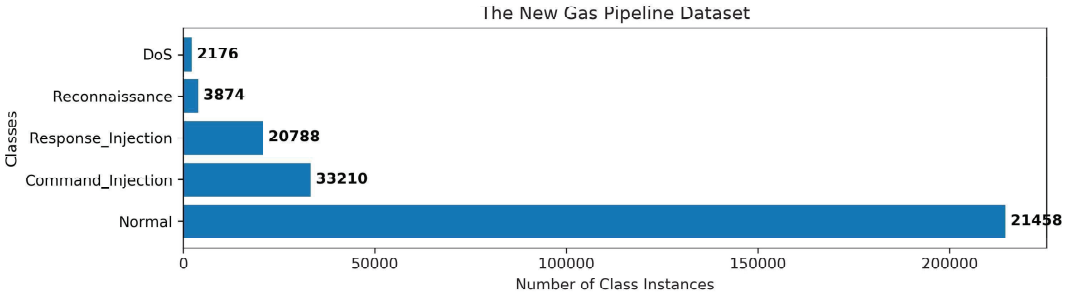


FIGURE 3.7: Four Main Attack Group and Normal Classes

- ***Command injection attacks*** - contains three attacks types, malicious state command injection (MSCI), malicious parameter command injection (MPCI) and malicious function code injection attacks (MFCI). These attacks inject control configuration commands to modify the system state and behaviour, resulting to:
  - *loss of process control*
  - *device communication interruption*
  - *unauthorized modification of process set points*
  - *device control*

- ***DoS attacks*** - disrupt communications between the control and the process through interruption of wireless networks or network protocol exploits.
- ***Reconnaissance*** - collects network and system information through passive gathering or by forcing information from a device.

3.5.2.3 Identified Cyber Threats in NGP Dataset

The original gas pipeline data as described in [252] was improved to create a new NGP data by:

- *parameterizing* and *randomizing* the order in which the attacks were executed.
- executing *all the attacks* as contained in the original data.
- implementing all the attacks in a *man-in-the-middle* fashion
- including all the *four types of attacks* as shown below
  - ***Interception*** - this type of attacks are sent to both the attacker and to the initial receiver. It enables gaining system information such as normal system operation, each protocols node, the brand and model of the RTUs that the system is using.
  - ***Interruption*** - attack is used to block all communication between two nodes in a system. Example: DoS between the MTU and RTU slave device in the gas pipeline.
  - ***Modification*** - attacks allows an attacker to modify parameters (set point parameter exclusively and leave all other parameters untouched) or states in a system, such as the gas pipeline.
  - ***Fabrication*** - in this type of attack, attackers execute this type of attack by creating a new packet to be sent between the MTU and RTU.

3.5.2.4 Raw Dataset

The raw unprocessed data contains six features within each instances of the raw data as illustrated in Figure [3.8].

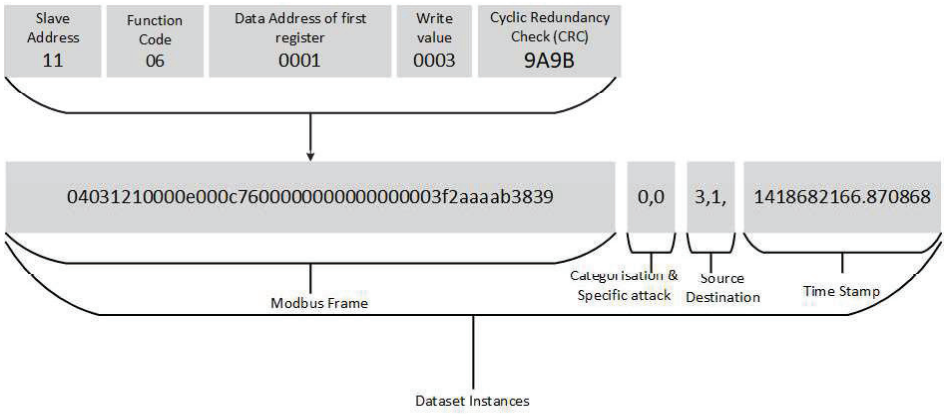


FIGURE 3.8: The instances within NGP raw dataset

- ***The first feature*** - represent the Modbus frame as received either by the master or slave device. All valuable information from the network, state, and parameters of the gas pipeline are also contained in this Modbus frame.
- ***The second and third feature*** **-** represent the attack category and specific attack that was executed. In case of Modbus frame normal operation, both of these features will report a zero. Both are useful to train a supervised learning algorithm, as they allow the algorithm to learn the behaviour of these attack patterns.
- ***The fourth and fifth features*** - represent the source and destination of the frame. There are only three possible values for the source and destination feature. The value can be a *'1'*, indicates that the master device sent the packet, *'2'*, meaning the man-in-the-middle computer sent the packet, or *'3'*, indicates that the slave device sent the packet.
- ***The last feature (6th)*** - contains a time stamp which can be used to calculate the time interval between change. In system normal operation, slight change may be observed between time intervals, however any modification or malicious activity such as malicious command injection may lead to noticeable time interval change.

3.5.3 UNSW-NB15 Dataset

UNSW-NB15 dataset as representation in Figure [3.9] and [3.10] was created by Australian Centre for Cyber Security (ACCS)[8] in their Cyber Security Lab. This hybrid of modern normal and abnormal network traffic features was created using the IXIA PerfectStorm tools[9] to simulate nine families of attack categories as follows: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. Each record contains 17 features in each network packet.

In other to identify an attack on a network system, a comprehensive dataset that contains normal and abnormal behaviours are required to carry out a proper evaluation of network IDS effectiveness and performance [240]. Hence, the UNSW-NB15 dataset was chosen for this study as the IXIA PerfectStorm tool used to generate the data contains all information about new attacks on CVE website[10], which is the dictionary of publicly known information security vulnerability and exposure and are updated continuously as stated in [253].

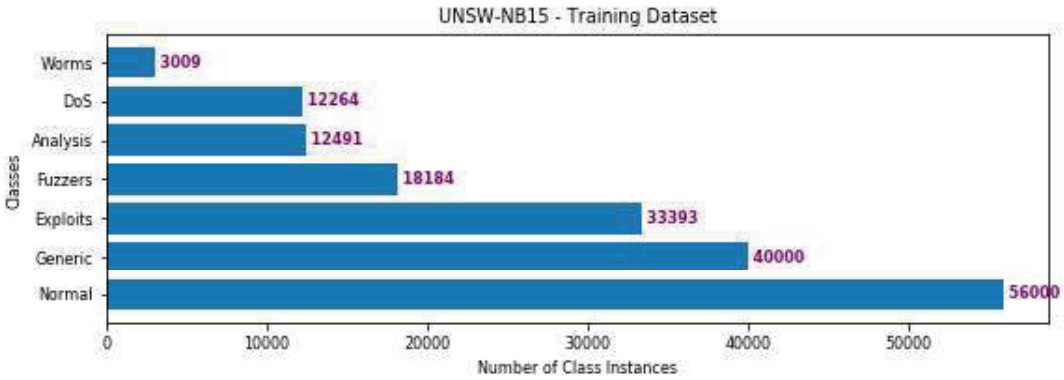


FIGURE 3.9: UNSW-NB15 train dataset

[8]ACCS: https://www.unsw.adfa.edu.au/unsw-canberra-cyber
[9]IXIA: https://www.ixiacom.com/products/perfectstorm
[10]CVE: https://cve.mitre.org/

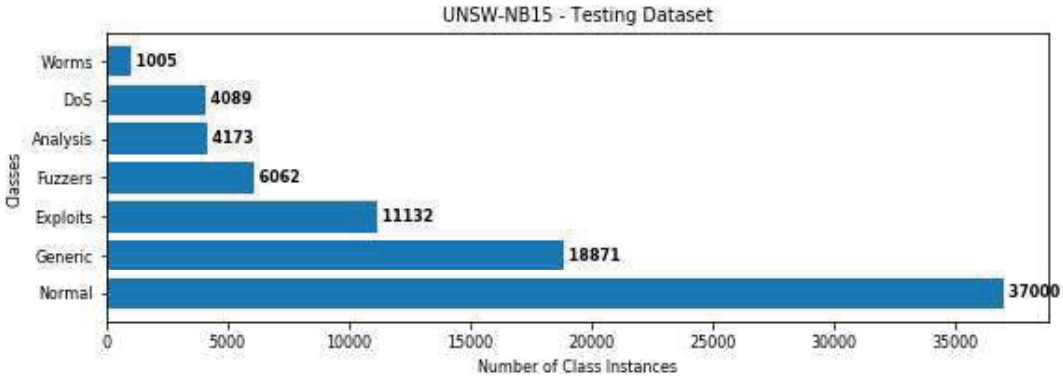


FIGURE 3.10: UNSW-NB15 test dataset

3.5.3.1 UNSW-NB15 Data Attack Type and Description

The UNSW-NB15 dataset is captured raw traffic data as Pcap files using Tcpdump tools. The data is partitioned into training and testing sets containing a total records of ***175331*** training and ***82332*** testing sets. Each set consists of 9 attacks families and normal events as described in Table [3.3].

TABLE 3.3: UNSW-NB15 Data Attack Type and Description

Attack Type	Description
Fuzzers	Attacker feed in massive input data into the operating system, program or network to make it crash in search of loopholes.
Analysis	Diverse intrusion that penetrate the web through emails (e.g., spam), ports (e.g., port scans) and web scripts (e,g., HTML files).
Backdoors	Stealthy way of securing unauthorized remote access to a device by bypassing normal authentication and locating the entrance to plain text.
DoS	An intrusion which causes the computer memory, to be extremely busy to prevent the authorized requests from accessing a device.
Exploits	A sequence of instructions that take advantage of a vulnerability glitch or bug to be caused by an unintentional or unsuspected behaviour on a host or network.
Generic	Technique that establishes against every block-cipher using a hash function to collision without respect to the configuration of the block-cipher.
Reconnaissance	Attack probe that gathers information about a computer network to evade its security controls.
Shellcode	Attacker penetrates a slight piece of code starting from a shell to control the compromised machine.
Worms	This type of attack replicates itself and spread on other computers using target computer network. It depends on the security failures on the target computer to access it.

3.5.4 KDDCup'99 Dataset

In the prospect of developing an effective IDPSs, many researchers have carried out different analysis on the KDDCup'99[11] dataset and the improved version NSL-KDD[12] dataset by employing different techniques and tools. Various statistical analysis with KDDCup'99 has also revealed some inherent drawbacks on KDDCup'99 from DARPA in [254, 255] as follows:

- ☐ It contains redundant records which can be removed to enable the classifiers to produce an un-biased result.
- ☐ The existence of duplicate records in test datasets.
- ☐ No exact definition of the attacks.

These drawbacks which may affect the accuracy of many modelled IDPSs have been refined and the outcome is the NSL-KDD dataset as proposed in [255].

This improved NLS-KDD dataset does not:

- ☐ include any redundant records in the train set; therefore, the classifiers will produce an un-biased result.
- ☐ contain any duplicate records in the proposed test sets; therefore, the learners' algorithm performance is not biased.
- ☐ the number of selected records from each difficulty level group is inversely proportional to the percentage of records in the original KDDCup'99 dataset.
- ☐ enough records are available in the train and test sets which makes the datasets reasonably sufficient to run an experiment.

Studies on the KDDCup'99 and NSL-KDD datasets for IDPSs analysis and network-based IDPSs performance evaluation do not reflect network traffic and modern low footprint attacks due to new evolving modern and hybrid attack as APTs [180, 253, 256]. [255] added that NLS-KDD may not be a perfect representative of existing real networks as this dataset still suffers from some inherent problems as discussed by [257].

However, due to lack of public datasets for NIDPSs, it is believed that it can still be applied as an effective benchmark dataset to help researchers compare different IDPSs techniques [255].

3.5.4.1 KDDCup'99 and NLS-KDD Datasets Limitations

Several studies including [255, 257, 258], highlighted that the KDDCup'99 and its improved version NSL-KDD datasets which are widely adopted benchmark datasets for IDPSs analysis and NIDPSs performance evaluation does not reflect a realistic output performance because of few limitations as follows:

Firstly, the KDDCup'99 training data set contains lots of redundant records affecting the results of detection biases learning toward the frequent records in [259, 260]. The second limitation is that there are multiples of missing records that are a factor in changing the nature of the data thereby eliminating classifiers biased to more repeated records [257]. However, considering the intent of the NSL-KDD dataset, it has resolved few issues such as missing values, duplicate records and imbalance between attack vectors among the normal and abnormal records in the training and testing phase to decrease the false

[11] KDDCup'99: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[12] NSL-KDD: https://web.archive.org/web/20150205070216/http://nsl.cs.unb.ca/NSL-KDD/

alarm rates (FARs) [255]. Also, this dataset does not contain a comprehensive representation of an emergent evolving footprint attack such as APT scenario which is the major disadvantage of NSL-KDD [255, 256, 257].

3.5.4.2 KDDCup'99 Dataset Statistical Analysis Representation

The full KDDCup'99 data contains **4898430** records of network traffic, a total of ***311029*** corrected data and a 10% subset data containing ***494021*** records with ***396,743*** packets associated with a cyber attack, see Table [3.4]. Each record contains 42 features in each network packet. The attacks are grouped into 4 different attack categories as can be seen in Figure [3.11] as follows: user to root (U2R), root to local (R2L), Prope and DoS.

TABLE 3.4: KDDCup'99 Dataset Attack and Normal Instances Distribution

Attack Categories	Attack Type	10 % KDDCup'99 Subset data	Corrected KDDCup'99 Test data	Full KDDCup'99 Train data	NSL-KDD data Train | Test	Description
Normal	normal	97278	60593	972780	67343 | 9711	Normal event connections records
DoS	back, land, neptune, pod, teardrop, smurf	391458	229853	3883370	45927 | 7458	Attack that obstructs and makes system services unavailable to legitimate users
R2L	multihop, ftp_write, guess_passwd, imap, phf, spy, warezclient, warezmaster,	1126	16347	1126	995 | 2754	Unauthorized access to a system from remote computer
U2R	loadmodule, buffer_overflow, rootkit, perl	52	70	52	52 | 200	Attackers acquire local super-user (root) privilege access on a specific computer or system
Probe	nmap, ipsweep, portsweep, satan	4107	4166	41102	11656 | 2421	Seeks and gathers information about the computer system and network of their target
Total		494021	311029	4898430	125973 | 22544	

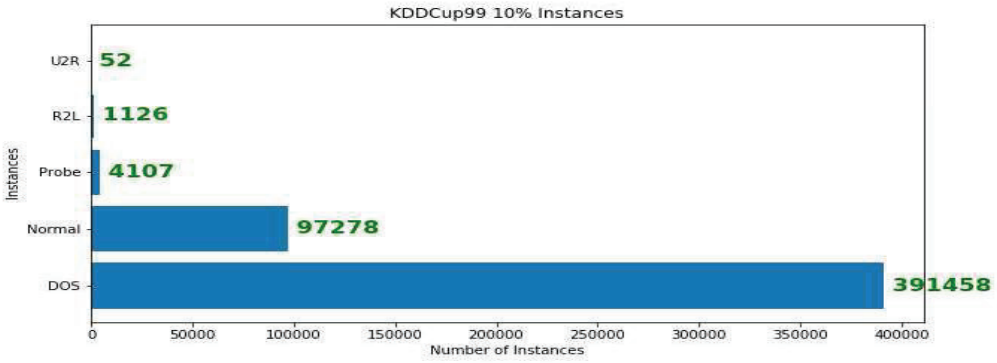


FIGURE 3.11: 10% KDDCup'99 Data Records Distribution

3.5.5 Comparison of Experimental Datasets

Table [3.5] contains nine parameters used to summarize the comparison among the NGP, UNSW-NB15 and KDDCup'99 datasets used for this thesis.

TABLE 3.5: Comparison of the four Experimental Datasets

Parameters	NGP Dataset	UNSW-NB15 Dataset	KDDCup'99 Dataset
Number of Records	Total of 274,628 records with 214,580 Modbus network packets and 60,048 packets associated with attack instances	Partitioned into 175331 Training and 82332 Testing sets	494021 (10% of total corrected records)
Attack families	7 attack families with 35 specific attacks type. See Table [3.2]	9 attack families. See Table [3.3]	4 attack families with 22 specific attacks types. See Table [3.4]
Number of distinct IP address		45	11
Number of networks		3	2
The duration of data collection	30 days	16 and 15 hours	5 weeks
Format of data collected	Attribute Relationship File Format (ARFF) and raw text (cvs) file	Pcap file	tcpdump, BSM and dump files
Simulation	Yes (Matlab Simulink – used to model gas pipeline, valves and pump, SimHydraulics Simulink package - used to model pipeline component	Yes	Yes
Feature Extraction tools	Wireshark and Snort – used to classify simulated network traffic as Modbus / TCP packaets Auto IT script (to allows direct interaction with iFIM HMI) and Datalogger	Bro-IDS tools, Twelve algorithms developed with c# programming language and Argus	Bro-IDS tool
Number of features extraction	17	49	42

3.6 Preliminary Experiments

The published preliminary experiments carried out over the course of this study which has formed part of this thesis are discussed in this Section.

3.6.1 Application of Machine Learning Algorithms for Threats Detection

In this experiment, the application of artificial immune system (AIS) and RNN variants for APT detection were explored. The result achieved indicated that the variants of the suggested algorithms were able to provide not only detection capability, but can also classify malicious data traffic with respect to the type of APT attacks as recorded in [51].

Threats to information and network security remain one of the biggest challenges facing organisations and industries at different operational levels. There have been a number of successful breaches of critical infrastructure. The WestRock packaging attack incident is an example of cyber attack leveraged by cyber criminals on manufacturing and other OT systems to cause damage to an organizational critical infrastructure This type of attack has drawn special attention to the possibilities of APT attacks on the ICS such as DCS & SCADA network. This has also led to a number of research interests in developing methods to detect and mitigate any attack within network and isolated devices.

The application of artificial intelligence (AI) inspired by AIS [261, 262], DL [263, 264] and ML algorithm[229] in APT intrusion detection has attracted more research attention. Security practitioners believe that these approaches may be a solution to APT and other cybersecurity issues.

Notwithstanding the popularity of the above mentioned techniques and extensive academic research, there is every need to exploit this knowledge of the system in detection of operational behaviour. Since sensors are been controlled in real-time similar to how the human body immune system detects and removes or destroys threats from numerous pathogens such as viruses, bacteria and parasites [265, 266]. Various techniques and models have been developed and applied in securing information system and critical infrastructures against APT and cyber threats.

However, to mitigate the vulnerabilities issues related to the mobile agent platform security usage lag in computational time, the authors in [266], applied AIS based model that will give the separation of duties and clones to handle multiple threats simultaneously as to achieve the computational efficiency. To implement this immune system functionalities, they have created artificial immune system components equivalent to that of a biological immune system. The pathogens represent the foreign mobile agents coming to the platform to perform some computation. Patterns are the antigens which are the functional codes of the foreign mobile agent. This will detect the malicious agent by extracting and matching it with the available malicious patterns. Moreover, an AIS can identify the new malicious patterns through monitoring the agent process. The experimental results achieved indicates that the model will consume less computational time when compared with the other approaches. Despite this experimental result, the implementations of AIS model with the capability of detecting malicious agents, while monitoring the process of foreign mobile agent in the sandbox, are yet to be carried out at the time of this write up.

Sim et al [267] have applied the combination of immune metaphor with genetic programming in their work "Network for Lifelong Learning (NELLI) system". NELLI is a first step towards developing Lifelong Learning Optimiser (L2O) systems [173]. NELLI has been applied to bin-packing and job-shop scheduling domains. In any given domain, NELLI independently generates a group of optimisation algorithms that has the capability of solving a diverse range of problem instances. Given a domain, NELLI has demonstrated that, to improve its performance when exposed to more instances that exhibit different characteristics from those previously exposed to, it generates an ensemble of optimisation new algorithms capable of solving various range of problem instances. Thus, since NELLI retains memory, it quickly returns new algorithm that exhibits good performance when re-exposed to any instances seen in the past [267]. However, the NELLI model has not been tested for APT detection in the cybersecurity domain.

Hence, adaptation of artificial immune system combined with integration of optimised continuous ML approach to detect attack instances offers a great potential to pregenerate algorithms in anticipation of future demand, thereby increasing the efficiency of the system. The author in [173], suggested for a shift towards the direction of optimisation community rather than focusing effort on developing more complex algorithms that is trained on large static datasets. A move towards developing systems that independently and continually generate specialised algorithms on demand may bear considerable fruit.

3.6.1.1 Experiment One

The purpose of the experiment is to examine the performance of two different approaches, the AIS-NSA and LSTM-RNN for APTs detection. Two tasks that involves LSTM-RNN model application using corrected 10% KDDCup99 dataset containing 494021 records were performed. The first task focused on deriving hyper-parameter values for best performance model. In the second task, the achieved hyper-parameter best values were applied in measuring the model performance. The output were compared to previously published AIS-NSA [208, 268] and [269] application as highlighted in Table [3.6]-[3.8]. The data was partitioned into 75% as training set and 25% as testing set for the binary and multi classification respectively, utilizing all the features. The training dataset were normalised from 0 to 1. This was trained using sigmoid activation function through time with ADAM optimiser, sigmoid function was used on all the three gates and, categorical & binary cross entropy as loss function for multi and binary classification respectively.

3.6.1.2 Results and Discussions - Experiment One

TABLE 3.6: Expanded Algorithms Names - the list of expanded names of all the algorithms as used for this experiment are shown on this table [51].

Algorithms (AG)	Expanded Algorithms Name
SVM	Support Vector Machine
KNN	k-nearest Neighbors
DTC	DecisionTree Classifier
RF	Random Forest Classifier
LR	Logistic Regression
ADB	AdaBoost Classifier
NB	Naive Bayes
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
AIS	Artificial Immune System
(NS-AFB)	Negative Selection with Antigen Feedback

■ **Confusion Matrix:**

The confusion matrix as represented in Figure [3.12] are the predicted and the actual true binary classifications of normal/attack and detection of all the four attacks group for each of the algorithms. The visual observation of the Figure 3.11, shows a clear picture of the number of instances of the R2L, U2R and Probes with lower connection records, while normal and DOS appear to have more connection records. Those group with more records are learnt properly without confusing their identity while those with fewer connection records during training did not show good true positive rate and precision as those appear to be difficult to easily identify. This indicates data imbalance problem. More records of "neptune" that belongs to DOS attack class, "satan" attacks belonging to Probe and "normal" are contained in the dataset with fewer records of other attack groups. The RNNs

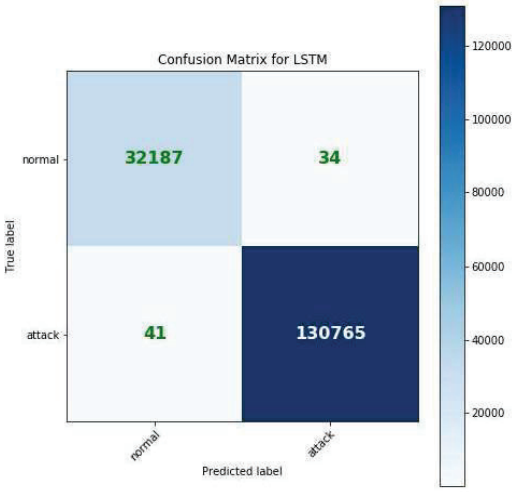
model was used as classifier and detector. As a binary classifier to separate normal from attacks instances, the RNN variants were able to achieve a significant result of 99.99% average accuracy. A closer observation of the individual performance of each of the RNNs indicates that LSTM model yielded a better result than the RNN with insignificant result. An outstanding overall performance of this model as indicated with good DOS attack detection and acceptable detection of Probe attack.

TABLE 3.7: Average Binary Summary Results - comparative summary result of the ML algorithms, the LSTM-RNNs network and the result of previously published AIS (NS-AFB to be more precise) [51].

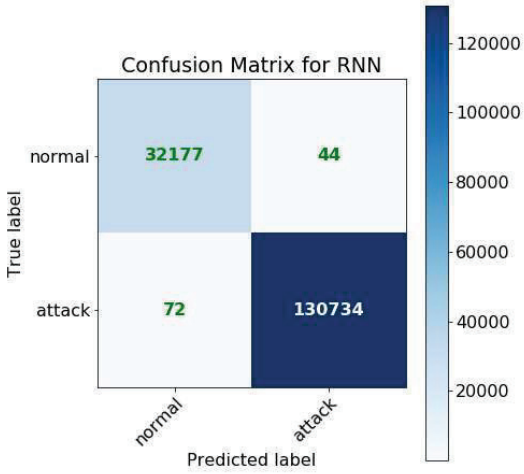
AG	DT	Acc	TPR	FPR	Prec	Rec	F-score
SVM	0.994	0.981	0.998	0	1	0.999	1
KNN	0.999	0.999	0.998	0	1	0.999	1
DTC	0.999	0.999	0.998	0	1	0.999	1
RF	0.999	0.999	0.998	0	1	0.999	1
LR	0.997	0.998	0.993	0.001	0.999	0.998	0.998
ADB	0.998	0.999	0.997	0.001	0.999	0.999	0.999
NB	0.945	0.945	0.784	0	1	0.932	0.965
LSTM	0.999	0.999	0.999	0	1	1	1
RNN	0.999	0.999	0.998	0.001	0.999	1	0.999
GRU	0.999	0.999	0.998	0	1	1	1
AIS [268] (NS-AFB)							
Attacks	0.952	-	0.998	0.479	-	-	-
Normal	0.992	-	0.998	0.790	-	-	-

TABLE 3.8: Summary Results of Average Multi-Class - this table contains the output summary of the performance of each of the algorithms in detecting the four main attack groups [51].

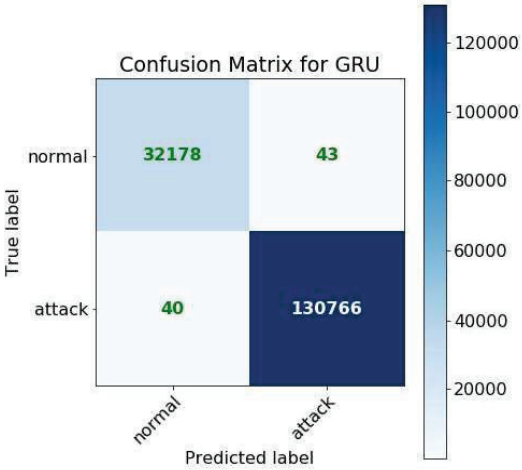
AG	DT	Acc	TPR	FPR	Prec	Rec	F-score
SVM	0.991	0.999	1	0	0.999	0.999	0.999
KNN	0.999	0.999	1	0	0.999	0.999	0.999
DTC	0.999	0.999	1	0	0.999	0.999	0.999
RF	0.999	0.999	1	0	0.999	0.999	0.999
LR	0.997	0.998	1	0.001	0.998	0.998	0.998
ADB	0.923	0.924	0.998	0.205			
NB	0.931	0.931	1	0.033	0.984	0.931	0.954
LSTM	0.999	0.999	1	0	0.999	0.999	0.999
RNN	0.999	0.999	1	0	0.999	0.999	0.999
GRU	0.999	0.999	1	0	0.999	0.999	0.999



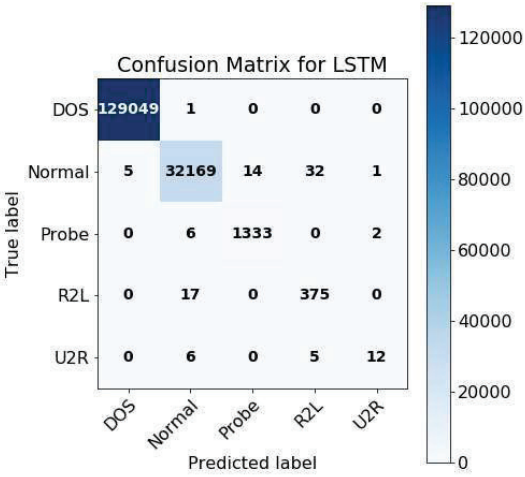
(A) Binary Confusion Matrix for LSTM



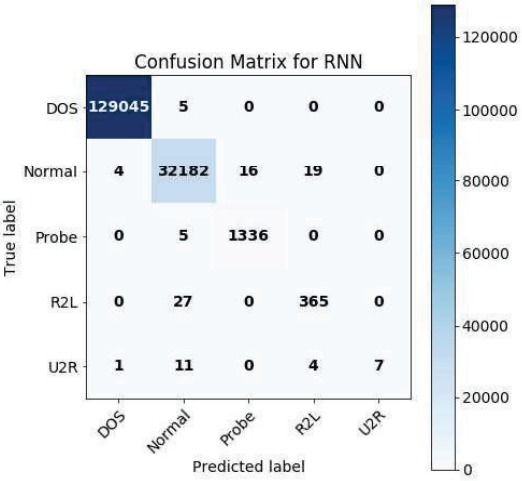
(B) Binary Confusion Matrix for RNN



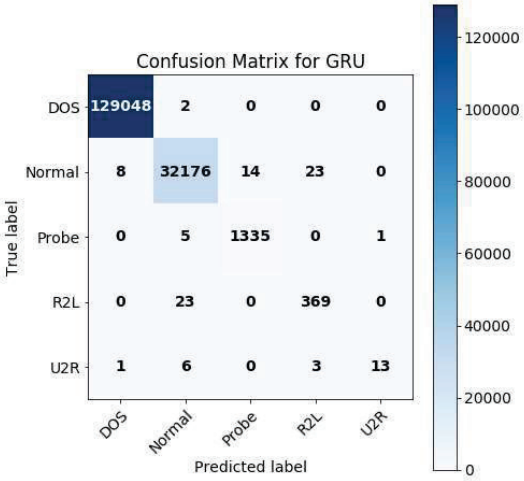
(C) Binary Confusion Matrix for GRU



(D) Multi-Class Confusion Matrix for LSTM



(E) Multi-Class Confusion Matrix for RNN



(F) Multi-Class Confusion Matrix for GRU

FIGURE 3.12: Binary & Multi-Classification Confusion Matrix for all the Implemented Algorithm [51].

■ **Summary - Experiment One:**

A model based on RNN variants were implemented and evaluated the effectiveness of RNN variants. The results were compared to previously published work on AIS-NSA [208, 268] and [269] application. A further implementation of attacks classification with seven different classifiers as contained in Tables [3.7]-[3.8]. The results from this classification were also compared to RNN variants results. It can be observed that most of the algorithms applied in this experiment achieved a competitive accuracy rate with insignificant FAR while few such as ADB and NS-AFB achieved a noticeable FAR, although NS-AFB achieved a good percentage detection accuracy of 95.20% on attacks, 99.20% on normal with TPR of 99.809% on both attack and normal classes. During the training, RNN variants seems to be suitable for classifying high-frequency attacks and also the low frequency attacks with lower confidence prediction of 62.50%, 56.20% and 37.50% for LSTM, GRU and RNN respectively on multi attack detection, while achieving a very significant average accuracy of 99.99% for LSTM, GRU and RNN on differentiating attacks from normal instances. The percentage accuracy of RNNs model achieved on this preliminary experiment shows that the LSTM model performed slightly better than GRU and RNN model especially in differentiating attack from normal instances. Overall, the result suggest that the LSTM-RNNs model is a good candidate for developing attack detection systems.

3.6.2 Handling Minority Problem in Threats Detection

Detecting rare attack is a multiclass problem, such as detecting multi-steps behaviour of APTs. Targeted APT attacks presents an increasing concern not only to government, but also to both cyber security and business continuity. Detecting the rare attack is a classification problem with data imbalance. This part of the preliminary experiment explores the application of data resampling techniques, together with heterogeneous ensemble approach for dealing with data imbalance caused by unevenly distributed data elements among classes with focus on capturing the rare attack. It has been shown that the suggested algorithms provide not only detection capability but can also classify malicious data traffic corresponding to rare APT attacks.

The ability of an intrusion detection system to detect every possibilities of an active attack on a system is a global security challenge. The dynamic and diverse nature of TTP used by attacker to implement a complete APT scenario attack has resulted to generation of unevenly distribution among examples of different classes. Hence, learning from imbalanced data poses notable challenges for ML algorithms as pointed out by authors in [270] and [271]. Handling imbalance data distribution in multi classification problem based on ensemble supervised learning and problem decomposition with cost-sensitive learning are still an active research area in ML community as highlighted by the authors in [272], [273] and [274].

Majority of the proposed works in managing imbalance data classification has led to a significant pool of solutions geared towards addressing both binary and multiclass imbalance problem as recorded by Weiss et al. in [275]. Most of these solutions where mainly for binary imbalanced problem [270]. Thus, there is every need for research direction towards developing reliable solutions to deal with multiclass cases. Implementation of different data resampling techniques in combination with heterogeneous ensemble learning approach were explored.

3.6.2.1 Experiment Two

The tasks carried out are summarised as follows:

- ☐ Implementation of various oversampling and undersampling approaches for managing binary and multiclass imbalance datasets with main focus on minority class in multiclass label on two datasets (KDDCup991 and UNSW-NB152).
- ☐ Analyses of the impact of those approaches that could be used to improve the results obtained, without proposing a new technique for handling imbalance data.
- ☐ Conducted series of experiments to; evaluate the impact of resampling imbalance data and ensemble deep neural networks to (i) accurately detect and classify an attack as abnormal and (ii) classify different types of attack into individual attack family.

■ **Learning from Multiclass Imbalance Data:**

Classification and prediction has become an important task for behaviour amd pattern recognition as these events are rarely observed in any given network. This makes the classification and prediction task to suffer from imbalanced data as stated by authors in [276]. The problem of imbalance data is not limited to security domain, but also applicable in detection of fraudulent transaction or call [277], text classification [278], and several other domains. Hence the need for a better way of managing this imbalance data that will yield a high detection rate including identifying the rare type of attacks.

Uneven distribution of attacks categories in classification tasks is referred to as the problem of imbalance data. This is a situation whereby instances of certain classes occur more frequently than others (the minority class). This makes it difficult for learning algorithms as mentioned in ([279]. Several studies have demonstrated that uneven data distribution is not the only factor that affects performance of diverse modelling classifiers as discussed by authors in [279], [242], [280], and [281]. Other factors also includes the number of minority class. This could result to not having enough data for training the model. Minority class forming small distributed groups which may leads to class separability problem. This is the main problem with minority class as highlighted by authors in [279] and [282].

Again, a high classification error may as well contribute to poor performance of the validation matrix implemented in any given problem doman [242] and [283]. Also, class overlap as highlighted by Vuttipittayamongkol et al. in [243], is also known to have a higher impact on the classification of imbalanced datasets than the dominance of the majority class. Although their proposed approach "new undersampling method that eliminates negative instances from the overlapping region" has only been tested on binary class [243].

As an example, let us consider the multi-steps APTs detection problem, in which the percentage of the different dynamically generated transactions steps of APTs scenario comparing to the legitimate transaction of 99.98% is very low, as limited to 0.02%. As the case of KDDCup99 dataset used in this study that contains four attacks categories with records of denial-of-service (DOS) as 391458, surveillance (Probe) as 4107, remote-to-local (R2L) as 1126 and user-to-root (U2R) as 52 of which R2L and U2R in essence very rare (see Figure [3.11]). However, any approaches which do not take into consideration the imbalance distribution of class elements, may lead to increase difficulty of the classification task as observed in the previous experiment [51].

During the training, it was observed that RNN variants appear to be more suitable for classifying high-frequency attacks. Also the low frequency attacks with lower confidence prediction of 62.50%, 56.20% and 37.50% for LSTM, GRU and RNN respectively on multi-class attack detection task, while achieving a very significant average accuracy of 99.99% for LSTM [51], although accuracy is not a recommended performance matrix for such task.

3.6.2.2 Results and Discussions - Experiment Two

The comparative overall average summary result achieved by individual algorithms before and after applying SMOTE oversampling techniques on both KDDCup99 and UNSW-NB15 datasets are shown on Figure [3.14]. RNNs variants model were used as classifier and as a detector. As a binary classifier to separate attacks instances from normal network events. This approach achieved a noticeable overall average accuracy of probability confidence & macro f1 results of (99.95% & 99.93% and 99.95% & 99.92% on KDDCup99) and (76.09% & 73.06% and 75.92% & 72.80% on UNSW-NB15) datasets with & without oversample technique applied. Comparing the result between oversampled and non-oversampled data, the difference is very insignificant.

Notwithstanding, since detecting the minority class of attacks is the focus for the experiment, the chosen performance metric measures were calculated to get a better understanding of the overall model performance using the confusion matrix in Figure [3.13. The performance of these algorithms in classifying individual classes as represented in Table [2] & [3] from [1], shows a slightly difference in result in detecting the minority class of interest in both resampled and non-resampled data for both used datasets. As a detector, the overall average detection rate improved from 63.20%, 68.40% & 57.60% to 78.90%, 78.90% & 73.70% for minority class of interest (U2R) in KDDCup99 dataset. In the case of detecting Worms in UNSW-NB15 dataset, although implementing SMOTE oversampling approach did make a huge impact as the overall average detection rate improved from 0.03%, 0.06% & 0.09% to 32.50%, 35.50% & 23.2%. However, the detection rate are still below average. This demonstrated that uneven data distribution is not the only factor that affects performance of various modelling classifiers as discussed also by authors in [280] and [279], high classification error highlighted by Sáez, et al [242]. These factors, contributed to the poor performance of the validation matrices implemented in this experiment.

In Figures [3.15] and [3.16], there is existence of some spikes in the validation accuracy and loss result. The individual model accuracy and loss per epoch, achieved training and validation accuracy of 99.96%, 99.95% (LSTM); 99.95%, 99.94% (RNN); and 99.96%, 99.95% (GRU) with overall validation average accuracy of 99.95% and average mean detection accuracy improving from 91.43% to 94.33% for oversampled and non-oversampled data. The accuracy result for the training and validation accuracy appear to be very close to each other, this indicates that the model is not overfitting.

■ **Summary - Experiment Two:**

In this experiment, a heterogeneous ensemble learning approach was implemented with application of SMOTE oversampling data resampling techniques on two different datasets. The impact of implementing resampling techniques and model performance on imbalance data for multi-class classification with focus on detecting minority class of interest were also evaluated. Further implementation and analysis were carried out to evaluate the ability of the RNNs variants to accurately detect and classify attack as abnormal, and how accurately this model will detect different type of attacks. The result obtained from resampled data was compared to the result obtained without application of any data resampling.

Considering the result recorded in Figure [3.14], most of the algorithms applied achieved a competitive overall average validation accuracy rate of 99.95% with 0% overall average validation loss for KDDCup99 dataset. At the same time achieving a maximum average mean accuracy of 81.02% with a noticeable validation loss for UNSW-NB15 dataset. Although the model demonstrated a good percentage detection overall average mean accuracy, precision, sensitivity and F1-score of 99.96%, and detection rate of minority class (U2R) improved from 63.20%, 68.40% & 57.60% to 78.90%, 78.90% & 73.70%.

Furthermore, with the percentage macro-average obtained from both datasets, these indicate that the classifier performs very well for each individual class. It can also be seen that the implemented model on experiment performed significantly better when dealing with KDDCup99 than UNSW-NB15 datasets, especially in overall performance validation. Nevertheless, the overall result suggest that high classification error and class separability problem of minority class has a noticeable impact on model overall performance with respect to application domain as seen with the case of KDDCup99 and UNSW-NB15 datasets.

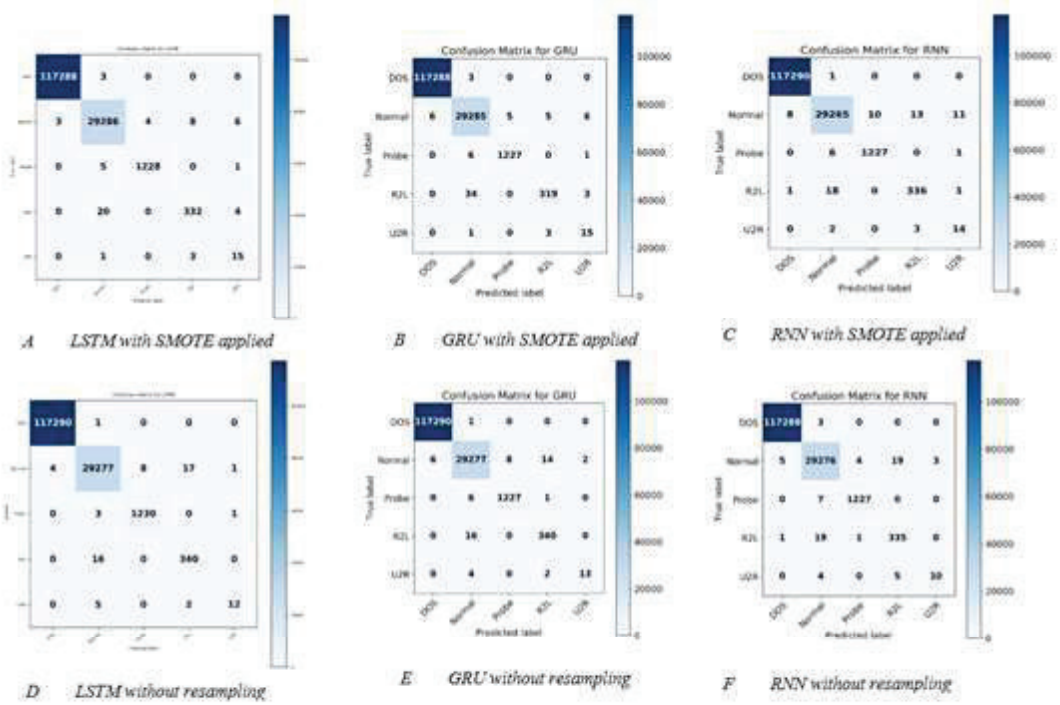


FIGURE 3.13: 10% KDDCu99 dataset confusion matrix of 5 classes with and without SMOTE oversampling techniques applied [1].

Comparative overall performance result table for multiclass classification data analysis						
10% KDDCup99 dataset						
Criteria	LSTM		RNN		GRU	
No of Samples	494020 - 10% KDDCup99					
	SMOTE	No-Re	SMOTE	No-Re	SMOTE	No-Re
TP	148149	148149	148132	148126	148134	148147
FP	58	62	75	71	73	60
TN	148149	148149	148132	148126	148134	148147
FN	58	62	75	71	73	60
OaAc (%)	99.96	99.96	99.95	99.95	99.6	99.96
ValLoss (%)	0	0	0		0	0
	Scores					
AgAc (%)	99.98	99.98	99.98	99.98	99.98	99.98
P (%)	99.96	99.96	99.95	99.95	99.95	99.96
Se / TPR (%)	99.96	99.96	99.95	99.95	99.95	99.96
Sp (%)	99.96	99.96	99.95	99.95	99.95	99.96
f1 (%)	99.96	99.96	99.95	99.95	99.95	99.96
Voting by Probability Confidence						
	SMOTE	No-Re				
OaAgAc (%)	99.96	99.95				
macro f1 (%)	92.93	91.24				
UNSW-NB15 dataset						
Criteria	LSTM		RNN		GRU	
No of Samples	175331 Training and 82332 Testing - UNSW-NB15 dataset					
	SMOTE	No-Re	SMOTE	No-Re	SMOTE	No-Re
TP	36333	43559	41199	34403	30716	36333
FP	45999	38773	41133	47929	51616	45999
TN	36333	43559	41199	34403	30716	36333
FN	45936	38779	3997	47935	51049	45936
OaAc (%)	48.63	52.91	50.04	41.79	37.31	41.13
ValLoss (%)	2.75	2.23	2.52	2.35	3.44	3.98
	Scores					
AgAc (%)	77.73	81.02	78.95	74.71	69.45	76.14
P (%)	48.63	52.91	50.04	41.79	37.31	41.13
Se / TPR (%)	48.73	52.9	50.74	41.78	37.57	41.16
Specificity (%)	48.63	52.91	50.04	41.79	37.31	41.13
f1 (%)	48.68	52.9	50.39	41.78	37.44	41.15
Voting by Probability Confidence						
	SMOTE	No-Re				
OaAgAc (%)	48.37	48.11				
macro f1 (%)	30.1	31.1				

FIGURE 3.14: Overall summary performance result of class classification of both resampled and non-resampled data for 10% KDDCup99 and UNSW-NB15 datasets [1]

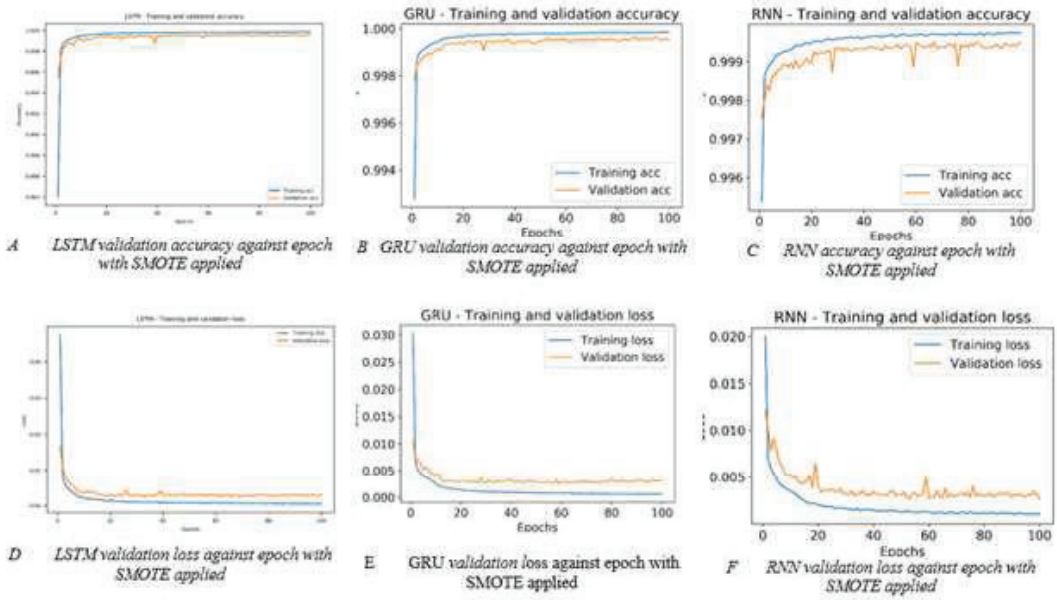


FIGURE 3.15: The visual representation of each algorithm's validation accuracy and loss rate on each iteration on 10% KDDCup99 dataset with SMOTE oversampling techniques applied [1].

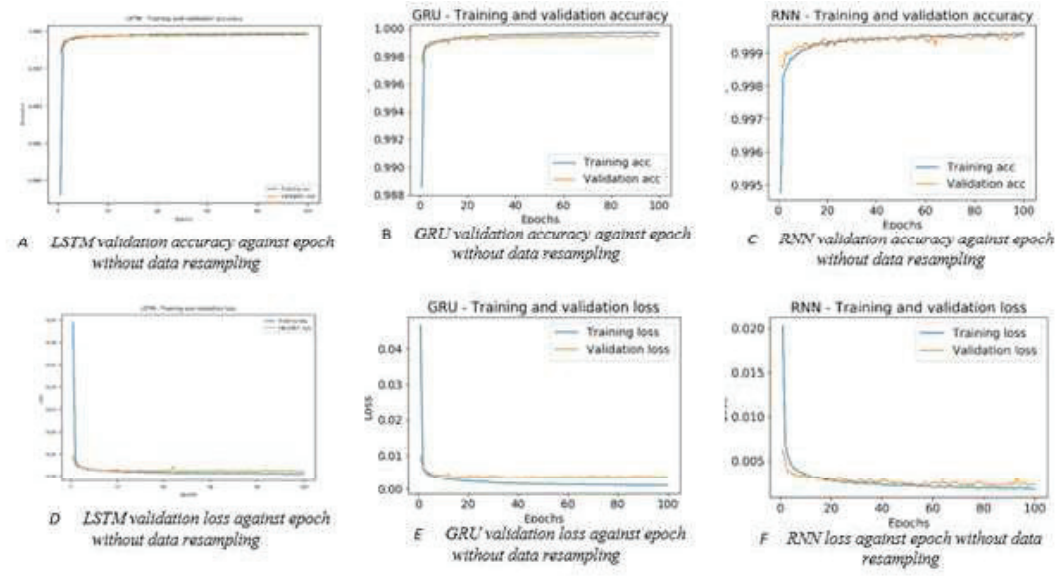


FIGURE 3.16: The visual representation of each algorithm's validation accuracy and loss rate on each iteration on 10% KDDCup99 dataset without resampling techniques applied [1].

Chapter 4

APT Detection Framework Based on APT Steps Analysis and Correlation

***

4.1 Rational of Design 81
4.2 APT_{DASAC} Architectural Design 82
4.3 The Three Stages of APT_{DASAC} 83
4.4 Summary 93

In this chapter, the description of the proposed APT detection system (APT_{DASAC}) framework for APT steps detection, designed and developed by the author, is presented. The architectural design of APT_{DASAC} is introduced along with a discussion of the main three stages of APT_{DASAC} as discussed in Section [4.3] are as follows:

- ☐ Data Input and Probing Layer,
- ☐ Analysis & Correlation Layer and
- ☐ Decision Layer stage.

4.1 Rational of Design

The rationale behind this thesis is based on few factors; firstly, considering that APT attack is a multi-step attack of which each attack step is delivered through the use of a particular attack tactics as briefly discussed in Sub-section [2.3.4.2] - *The MITRE ATT&CK Framework*. The detection of any single step of this attack tactics does not imply detection of an APT scenario [1]. Thus, to detect and mitigate an APT attack, the detection system should be able to go through the detection of each of the tactics or techniques used within each step of the APT lifecycle, as highlighted in Sub-section [2.1.3] - *An Overview of APT Lifecycle*. Hence, the detection model should be designed and developed to have the capability to detect most commonly used tactics to deliver individual APT attack steps at its early stage. This could prevent the attacker from achieving their goal of data exfiltration, and even curtail the damage that may have occurred had it not been discovered early enough.

Secondly, the model should be developed in such a way that it will be able to correlate the individual attack techniques and link the outputs of the individual detection modules (representing one step), thus reducing the chance of generating false positive rate of the detection system. Again, having reviewed some of the existing proposed model for APT detection, majority of these model focus on detecting a particular APT attack step. Few examples of such models are Active Learning Base [38, 224], APT-BN [235], without considering that a single attack step does not make up an APT attack campaign.

Nevertheless, it worth noting that some of these tactics that could be used during APT campaign might not necessarilly be an attack on it's own, as it could be a benign event (eg. port scanning - reconnaissances), techniques used for other type of attack such as Botnet attack [284]. Others could even be techniques such as Tor network connection used legally to protect system user's network traffic confidentiality which can as well be used for data exfiltration in APT attack, [248].

4.2 APT_{DASAC} Architectural Design

The architectural design of the proposed model for APT intrusion detection system (IDS) is built to run through three stages. This involves implementing a multi-step security detection approach that takes into consideration the distributed and multi-level nature of the ICS architecture and reflect on the APT lifecycle for the four main SCADA cyber-attacks as suggested in [52]. The output of the implementation of this APT_{DASAC} framework on NGP data, produces four detectable steps of attacks. An APT detection system should be able to detect every single possible step applied by an APT attacker during the attack campaign.

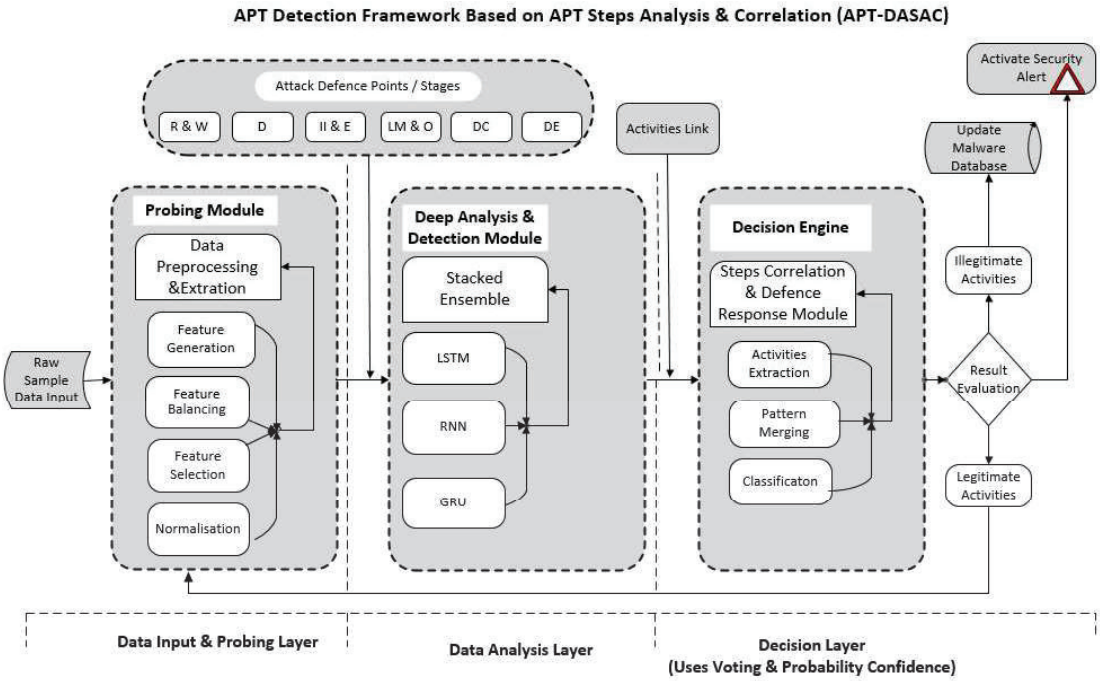


FIGURE 4.1: APT detection framework based on APT steps analysis & correlation (APT_{DASAC})

4.3 The Three Stages of APT_{DASAC}

The three stages of this model design as shown in Figure [4.1] are discussed as follows:

Proposition 1: The Three Stages of APT_{DASAC}

Stage 1: Data input and probing layer

Stage 2: Data analysis & Correlation layer

Stage 2: Decision layer

For the discussion of the proposed framework in this thesis, the NGP[1] and UNSW-NB15[2] datasets will be used for explanation and illustration of these three stages of the APT_{DASAC} framework as well as the specific step-by-step pseudocode for APT_{DASAC} and the detection process as described in the following Sub-sections.

The first stage of this approach is *"Data input and probing layer"*. This stage involves scanning the network traffic and processing the network activities so as to determine or detect any of the TTPs used during APT lifecycle. Given a certain time window, the data generated from the monitored network is processed by transforming the data into an appropriate data format to be utilized in the second stage which is *"Data analysis & Correlation layer"*. At this stage, the processed data is analysed. The output is an alert that represent one step of APT lifecycle. The activated alerts are fed to the correlation framework. This correlation framework processes the alerts as to filter some repeated alerts, cluster those alerts that are likely to belong to the same APT attack scenario, and then evaluate the degree of correlation between alerts of each cluster. The final stage applies the core process of APT_{DASAC}, which uses stacked recurrent neural network (RNN) variants to learn the behaviour of APT steps from the sequence data. These steps reflect the pattern of APT attack steps. In this stage *"Decision layer"*, the ensemble RNN variants is utilized to integrate the output and make a final prediction result.

- ***Step-by-Step Pseudocode for APT_{DASAC}* Layers:** The experimental implementation pseudocode of the proposed framework - APT_{DASAC}, Figure [4.1] is illustrated with Algorithm [1-4] as used to build the proposed model.

4.3.1 Data Input and Probing Layer

Data input and probing layer consists of two modules; (i) Data input and (ii) Probing module. The pseudocode for this module process is illustrated with Algorithm [1].

Data gathering and pre-processing is an important fundamental process in data analysis to gather and convert raw data into a usable dataset. There are a good number of different pre-processing approaches, these includes but not limited to noise reduction, data cleaning, outlier removal and removing/replacing missing data. None of these approaches are simple tasks to accomplish, but are very useful. The presence of noise may lead to model overfitting as a result of increase in the number of parameters due to the depth of the Model. For example: DNN as used at some point on this study are prone to this type of issue, where also a model that performed well on training dataset suddenly may perform very poorly on a test dataset.

[1] NGP: https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets
[2] UNSWNB: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/

At this stage, the *process_data module* takes raw network traffic data as an input from a specific problem domain, to process and transform the data into a meaningful data format that the algorithm requires by converting any symbolic attributes features and categorical features into usable feature, deals with null values using *steps 1-7b*, highlighted in Algorithm [1]. The output from this stage is a new transformed data (see Figure [5.1]) containing valuable information that the analysis/correlation stage will utilize.

Proposition 2: Data Pre-processing

The ***pre-processing data*** stage takes raw network traffic data as an input from a specific problem domain, processes and transforms the data into a meaningful data format that the algorithm requires by converting any symbolic attributes into usable features, and deals with null values using ***steps 1-7c in Algorithm [1]***. The output from this stage is a ***new_transformed_data*** containing valuable information to be utilize at analysis & correlation stage (see: Algorithm [5] and Code Snippet [C.7] for ***"process_data module"***).

- ***Data Input*** - involves data gathering, raw sample/simulated synthetic data been introduced into the system and transfering the collected data to probing module.
- ***Probing Module*** – involves data pre-processing and feature transformation which runs through four stages. Here all the data that has been collected and introduced into the module are encoded into numerical vector by the pre-processor ready to go through the neural network.
  - ***Feature Transformation***: UNSW-NB15 dataset consists of 42 features with three of these features been categorical *(proto, service and state)* data. These three features need to be encoded into numeric feature vectors as it goes into the neural network for analysis, classification, detection and prediction. For this reason, the Pandas $get\_dummies()function$ (see: Code Snippet [C.2]) was utilized. This function creates new dummy columns for each individual categorical feature, this leads to increase in the number of columns from 42 to 196 features available for onward analysis.
  - ***Balancing Training & Testing Data Features***: This function is only invoked based on the nature of the features contained in both training and testing data of the domain network traffic information in use as the case of UNSW-NB15 dataset. Both training and testing data contains different number of categorical features, this implies that $get\_dummies()function$ will generate different number of columns for training and testing data. However, the number of features in both sets need to be the same. In this case, *set().union() function* were deployed to balance the training and testing datasets (see: Code Snippet [C.3]).
  - ***Normalization***: At this stage, the *ZScore* method of standardisation is used to normalize all numerical features to preserve the data range, introduce the dispersion of the series, and to improve model convergence speed during training (see: Code Snippet [C.4]).

    The Mathematical representation of *ZScore* is shown in Equation (4.1).

$$Z = \frac{x - \mu}{\sigma} \tag{4.1}$$

where Z is *ZScore*, x is the individual data point, μ is the mean of x for a given attribute that is uniform real number between 0 and 1, represented as Equation (4.2).

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.2}$$

and Rho (σ) is the standard deviation, represented in Equation (4.3).

$$\sigma = \sqrt{\frac{\sum (x-\mu)^2}{n}} \tag{4.3}$$

The categories *"attack_cat_column"* are all strings, these categorical data also need to be converted into one hot labels. Sklearn's *Label_Encoder()* from preprocessing library was utilized to encode these strings into numerical values (see: Code Snippet [C.5]).

Algorithm 1: Data Input and Probing Layer Pseudocode

```
- Pseudocode for Data Pre-processing
Get Raw Dataset                                   /* gather raw data */
while Get data do
    Step 1: Input the sample dataset
    Step 2: Convert the symbolic attributes features
    Step 3: return new set of data
    Step 4: Separate the instances of dataset into classes (y)
            (such as
              Normal, Naïve Malicious Response Injection (NMRI)
              Complex Malicious Response Injection (CMRI)
              Malicious State Command Injection (MSCI)
              Malicious Parameter Command Injection (MPCI)
              Malicious Function Code Injection (MFCI)
              Denial of Service (DoS), and
              Reconnaissance (Recon))
          Step 4a: Encode the extracted categorical classes text values to indexes
    Step 5: Scale & Normalize data (x_t) into values from
            [0 − 1] using Equation [4.1]
    Step 6: Split dataset into training and testing data
    while Step 7: Training & Testing sets do
        Step 7a: Balance & reshape the training & testing
                 data features using Code Snippet [C.3] (Balancing
                 Data Features)
        Step 7b: return transformed new balanced & reshaped training
                 & testing data
        Step 7c: Pickle transformed training data into a byte
                 stream and store it in a file/database (.pki)
    end
    return (processed testing data)
    return (processed pickled training data)
end
```

4.3.2 Analysis & Correlation Layer

At this layer, the *"Pseudocode for Sequence Data Training, Validation and Testing"* from Algorithm [2] is utilized. The pre-processed data or the network traffic is scanned and processed to detect possible attack tactics (steps) used within a given time window during an APT lifecycle. Two difficulties were encountered; (i) class distribution problem and (ii) classification of rare attacks.

The rate of attack detection is affected by the parameters used as these parameters have direct impact on attack detection. Based on this, several experiments with different network configuration were implemented to find the best optimal values for parameters such as number of epochs, batch_sizes, dropout, learning rate, network structure etc.

Algorithm 2: Analysis Layer Pseudocode

```
- Pseudocode for Sequence Data Training, Validation & Testing
During the training and testing stage, steps [8a–8e] are followed in each iteration
begin
    Get Pre−Processed Data      /* analysis of train & test data */

    while Train Model do
        while Step 8: Train the model with the new training dataset do
                Step8a: Sequentially fetch a sample data (x_t)
                        from the training set
                Step8b: Estimate the probability (p) that the
                        example should be used for training
                Step8c: Generate a uniform random real number μ
                        between 0 and 1
                  Step8d: If (μ < p), then use x_(t) to update the RNN for
                        any training sample (x_i y_i)
            while in Step8 do
                Step 8e by
                repeat
                    Steps [1–4] of Algorithm [1]
                until there is no sample left in the training set
            end
        end
    end

    Step 9: Test model with testing data from Step [7b] in
            Algorithm [1]

    Step10: Compute and evaluate the Model performance
            accuracy output - classification, detection and
            prediction
    return (classification result)
    return (prediction result)
end
```

Again, considering these identified issues, to achieve a good detection rate for rare attack steps whilst maintaining overall good model performance, these two issues need to be carefully considered - the rare attack class distribution and the difficulty of correctly classifying the rare class. Firstly, when considering the class distribution, more emphasis should be placed on the classes with fewer examples. Secondly, attention should also be given to examples that are difficult to be correctly classified.

Taken these identified issues with class distribution and classification of rare attacks into account, the processed data are used to build the model that analyses and distinguishes real attack(s) activities from normal network events. This phase of the framework uses the six steps shown in Table [4.1], and also discussed in Sub-section [4.3.3] (APT_{DASAC} Attack Defence Points). These six steps have been identified as the main steps of APT lifecycle for this thesis to derive the detectable steps for each individual module shown in Table [4.2]. Detecting any of these six steps, does not infer detecting APT campaign. Hence,

each individual step detected should be merged together to construct a clear representation of an APT attack scenario. Thus, a correlation framework is required to connect the outputs of each individual attack defence point module. This process will help to reduce the FPR of the detection system. The output of this layer is passed to *"Decision Engine layer"*.

TABLE 4.1: APT_{DASAC} Detectable & Non-Detectable Steps of APT Lifecycle

Steps	APT steps	Detectable and Non-Detectable steps
Step 1	Reconnaissance and Weaponization	This stage involves information gathering about the target through monitoring and port scanning of the network traffic and building. This cannot be detected as it happened outside the target network.
Step 2	Delivery	The weaponized payload is executed/delivered through diverse means: USB sticks Email – executable file Website – malicious file download (SHA1, SHA256 and MD5 hashes) - malicious url Domain – Connection to any malicious domain name.
Step 3	Initial Intrusion and Exploitation	Detection of malicious IP address Detection of malicious Secure Sockets Layer (SSL) certificate Domain flux detection; attack eg. Fuzzer
Step 4	Lateral Movement and Operation	Here, the attacker has now established communication within the target system. The APT_{DASAC} at this point, constantly monitors the internal targets network traffic, including the inbound and outbound traffic movement, thus no form of detection occurs here.
Step 5	Data Discovery and Collection	Data encryption Move sensitive data Establish staggering server point, thus Scanning detection occurs here
Step 6	Data Exfiltration	Reverse shell or C2, to ex-filtrate the data, Onion routing protocol Initiate external connection. Establish C&C channels. Establish drop off server. - Tor connection detection

Full_APT_Scenario: Given a specific correlation time window, a full APT scenario is one in which all the possible detectable steps of an APT attacks are detected and correlated together. Thus, the four detectable steps of APT are detected. Lets consider APT lifecycle, Table [4.1] and detectable steps in Table [4.2], that contains all the possible detectable alert in APT_alert_dataset. The correlation module is able to detect all these steps, filters repeated alert previously seen, or which may not belong to the cluster group, and evaluate the degree of relationship between these steps. The full APT scenario can be expressed as Equations (4.4) - (4.8).

$$A = [aa \vee ab \vee ac] \tag{4.4}$$

$$B = [ba \vee bb \vee bc] \tag{4.5}$$

$$C = [ca] \tag{4.6}$$

$$D = [da] \tag{4.7}$$

$$Full\_APT\_Scenario = [A \wedge B \wedge C \wedge D] \tag{4.8}$$

TABLE 4.2: APT lifecycle detectable attack steps / alert as contained in APT_alerts_dataset.db, and also discussed in Sub-section [2.1.3].

APT Steps	Detectable Alerts
Step A (Entry Point)	disguised_exe_alert (a1) hash_alert (a2) domain_alert (a3)
Step B (Exploitation - C&C communication)	ssl_alert (b1) ip_alert (b2) domain_flux_alert (b3)
Step C (Asset/Data discovery)	scan_alert (c1)
Step D (Data Exfiltration)	tor_alert (d1)

4.3.3 APT_{DASAC} Attack Defence Points

In order to realize the great functionality of the APT_{DASAC}, this unit - APT_{DASAC} attack defence points contributed to an important part of it's functionality. The APT_{DASAC} design and implementation focused on six stages as to align with APT lifecycle discussed in Sub-Section [2.1.3]. This is to detect different TTPs highlighted in many reports [285, 286], and conference papers [11, 50, 83], in other to construct a clear picture of APT attack scenario. Taking into consideration the APT steps, mentioned in Sub-Section [2.1.3], and also summarised in Table [4.1] to show the APT_{DASAC} attack defence points for each individual APT step.

4.3.3.1 Step 1 - Reconnaissance and Weaponization (R&W)

This involves more of passive activities that focuses on information gathering about the target through social engineering techniques, monitoring, port scanning of the network traffic and OSINT tools. Most of these activities cannot be detected since it happened outside the target network.

4.3.3.2 Step 2 - Delivery (D)

This module is designed to focus on detection of executable file, malicious file download and connection to malicious domain name. The intelligent information gathered about the attacker's target is used to craft a payload to execute their exploits. The attackers use social engineering such as spear phishing which is the most commonly used technique to achieve the initial point of entry in APT campaign by sending phishing email to the target. Attackers sometimes may also compromise a trusted third party vendor, or frequently visited website by target, and use these to deliver and execute an exploit.

The spear phishing emails usually contain either a hyperlink to malicious file on the email body or a malicious executable file attachment. The email subject and text within the email body are usually crafted to something that are usually relevant to the recipient within the targeted environment/organisation. The executable files extension is ***".exe",*** usually made to look like a document with files extension such as ***.docx, .doc, .ppt, .pdf, .excel***, so that the victim will believe that it is a genuine normal document file extension and click on it.

At this point, the defence module detects if the content of the attached file is an ***.exe files*** over the network connections.The network traffic is processed, all the connections analysed and, checked to see if it is a disguised executable file with extension such as ***.exe*** in other to raise alert on delivering ***.exe file*** detection. Alert raised at this stage is *"disguised_exe_alert"*. The generation of *disguised_exe_alert* is illustrated with Algorithm[3]

This module also, checks and detects any malicious file that has been downloaded by one of the network hosts based on a blacklist of malicious file hashes on database [66]. The network event traffic is processed, and all network connections analysed. The SHA1, SHA256 and MD5 hashes are calculated for every identified file when transferring over network connection. The result of the calculated hashes are matched up with the blacklist database. It also checks and detects connection to a malicious domain name by filtering and analysing all the DNS traffic requests based on blacklist. The results are then matched up with DNS blacklist. Alerts triggered at this stage are *"hash_alert & domain_alert"*

4.3.3.3 Step 3 - Initial Intrusion and Exploitation (II&E)

This module is designed to detect infected connections such as malicious IP address, malicious SSL certificate by checking the connection between an infected host and a C&C server based on a blacklist of malicious IPs and SSL of C&C servers as highlighted in [287, 288, 289, 290, 291]. After processing the network traffic, all the secured connections are filtered. The SSL certificate associated with each secure connection is matched with the SSL certificate blacklist to determine if an alert can be raised. Alerts triggered at this stage are *"ssl_alert & ip_alert"*.

The domain flux technique is a common technique used for C&C communications. Each infected machine uses a domain generation algorithm (DGA) to generate a list of domain names [292, 293]. The infected host query's and connect to these great deal of generated domain as to link the infected host to the C&C servers. Domain flux attacks occur as a result of DNS query failures due to the infected host query for the existence of a large number of generated domains, where only one domain name has been registered. Analysing failed DNS query against the threshold for DNS query failures from the same IP address, will enable the detection of domain flux attacks, thereby identifying infected hosts, thus raising *"domain_flux_alert"*.

4.3.3.4 Step 4 - Lateral Movement and Operation (LM&O)

At this module, the attacker has already established communication within the target system. The APT_{DASAC} at this point, monitors the internal target network traffic, including the inbound and outbound traffic movement. Thus no form of detection occurs at this stage as this is happening within internal target network traffic.

However, the authors in [83], proposed a new approach named "HetGLM", for detecting LM attack during APT campaign. The authors constructed a heterogeneous graph with various network entities such as devices, users, processes, etc. This is a graph neural network (GNN)-based anomaly link detection algorithm, aimed to spot lateral movements

Algorithm 3: Pseudocode for Delivery Point Attack Detection

```
1  Step 2 - Detectable APT attack step
2  Begin
3      Get exe_file cluster table                    /* disguised_exe_alert */
4
5      Get new_event_connection                  /* new network connection */
6
7      Given file_name ← file name
8      Check
9      if the connection is known to the monitored network then
10         if the mail extension type is in exe_file cluster table then
11             if the mail extension type is file_name extension then
12             if the same disguised_exe_alert has previously been generated then
13                 goto End
14         else
15             if Generated new_event_alert disguised_exe_alert then
16                 Write the generated new_event_alert disguised_exe_alert into
                    disguised_exe_detection.log
17                 If the same alert is detected within the a given time_window,
                    suppress new_event_alert
18                 end if
19             else
20                 goto End
21             end
22             else
23         end
24     else
25     end
```

within a given network. The author reported that this approach was able to detect LM attack using public dataset. However, the validity of this model performance in real time scenario are yet to be confirmed.

4.3.3.5 Step 5 - Data Discovery and Collection (DC)

During this module, the scanning attack occurs, and important data discovered is encrypted. These actions are achieved by utilizing the privileged users' credentials to create redundant copies of C&C channels at several staging points, where the gathered information is packaged and encrypted before exfiltration in preparation for any change in security configuration [289]. Usually, the SSL encryption is used to secure communications in C&C, this makes it difficult to identify malicious traffic.

Through network scanning, important data such as closed, open & filtered ports, gateway filtering, firewall rules, router, IP address, MAC address, etc. can be obtained [223, 294, 295]. These information helps the attacker to ascertain which part of the system of interest for further exploitation. This type of attack can be detected, given a specific time interval, by tracking all attempted failed connection, and then matched with the set threshold for those failed connection attempts as to identify infected hosts. The alert raised at this stage is *"scan_alert"*.

4.3.3.6 Step 6 - Data Exfiltration (DE)

This module is designed for data exfiltration detection. The TOR communication network connection used during APT attack campaign for data exfiltration is an anonymous communication network used legally to secure and protect the privacy of user network traffic through network connections encrypting [90, 248, 249, 250].

The attacker uses TOR communication network connection remotely to instruct and direct infected machines [296]. The Tor achieves this by utilizing onion routing to direct client's traffic over a circuit of different relays to its destination, making it impossible for each single relay to know the complete path of the traffic [297]. The network traffic is processed and analysed, then the source and destination IP addresses for each network connection are matched with publicly published Tor servers list [298]. The alert raised at this stage is *"tor_alert"*.

4.3.4 Decision Layer

This Layer operates in three phases; firstly, it receives information from the analysis layer and extracts the attack step present. Secondly, it processes this information and links it to any previously identified attack steps that are related. Then, lastly, it uses voting and probability confidence to establish if the attack is a potential chain of attack campaign is found, and if it is consistent with other attack campaigns.

At this point, the attack impact is determined at this stage through the decision engine by correlating the output from the analysis layer using probability confidence to check for any presence of security risks. If an attack or security risk is present, it requests the defence response module to raise a security alert. This is checked with the previously detected step to see if this could be related to the newly discovered security risk alert. This is to reconstruct APT attack campaign steps, and hence highlight an APT campaign scenario if any, so that an appropriate action can be taken.

The impact of an attack can be considered as low depending on the attack activity stage. However, if this stage can be linked with other attacks step to show that it is part of that attack campaign, forming a full APT step cycle, then the impact at this stage can be considered as high. With this information in mind an appropriate response can be taken.

Algorithm 4: Decision Layer Pseudocode

```
- Pseudocode for Analysis, Detection and Prediction
In analysis detection & prediction stage, steps 11–17c are followed in each
  iteration
Get the processed Data      /* pre-processed data through Algorithm */
                                                  /* [1] is required */
while Get processed data do
    Step 11: Set ip_units, lstm_units, op_units and optimizer to define network
    Step 12: Fetch the processed data (x_t)
    Step 13: Select specified training_window_size_(tws) & arrange (x_t) accordingly
    while Step 14a: For n_epochs & batch_size do
        Step 14b: Take the input vector within specified
                  training_window_size_(tws) at time_(t)
                  together with previous information, initially set to 0
        Step 14c: Train the Network (L) x_(tws+1)
    end
    Step 15: Run Predictions using L
    Step 16: Calculate the categorical_loss_function (L_(o,y)) using Equation (2.10)
    while Step 17: Output & Result do
        Step 17a: The percentage detection rate of individual detected attacks
        Step 17b: return overall detection rate
        Step 17c: Confirmation if there is any existence or
                  complete APT steps campaign (full APT scenario)
    end
    return (prediction results)
    return (detection results)
end
```

4.4 Summary

The architectural design of APT_{DASAC} framework has been discussed in this chapter, together with the rationale behind this thesis. The APT_{DASAC} is partitioned into three main stages: the (i) data input and probing layer, (ii) analysis & correlation layer and (iii) decision layer.

The first stage, “data input and probing layer”, is the information gathering and data processing phase. This phase aims to transform the acquired information into an appropriate data format for usage in the second stage. The second stage, "analysis & correlation layer", utilized the six steps as identified as the main steps of APT lifecycle for this thesis to derive the detectable step for that particular module. This second stage uses the processed data from the first stage to build the model that aims to analyse network activities, filter different detectable alerts, correlates these alerts into it’s respective cluster. The result of this layer is passed to "Decision engine layer". This last stage, the decision layer module takes the output from the analysis & correlation layer using probability confidence to build a prediction model for APT steps detection, and then distinguishes attack from normal network activities, thus highlights an APT campaign scenario.

Chapter 5

Implementation of APT_{DASAC}

***

5.1 Implementation Setup 95
5.2 Hyperparameters Settings Used 96
5.3 Implementation of the Detection Framework 97
5.4 Detection Module 97
5.5 Summary 105

In this chapter, the description of the platform and the approach taken to implement the APT_{DASAC} framework will be introduced. The implementation setup, the hyperparameters settings used, the programming language and the tools used will be highlighted. Also, the implementation algorithms of each stage of the model will be presented.

5.1 Implementation Setup

The purpose of this implementation chapter is to examine the performance of implementing APT_{DASAC} framework as described in Chapter [4] for APT attack steps detection. This is to address some of the research questions in Chapter [1], especially *"RQ3"* of Sub-section [1.4.1], thereby deliver the thesis objectives *"RO1-RO3"* as highlighted in Sub-section [1.4.2].

There are varieties of widely used programming languages that could be utilized to create and develop a model in ML. Some of these languages are R, Python, C/C++, Java, MATLAB, and JavaScript. Python and R seem to be the mostly used languages in recent studies at the time of this study and writing this thesis. For this study, Python programming language were used in the implementation of all the modules to achieve it's functionalities on Jupyter Notebook environment as the interactive platform on GPU-enabled TensorFlow network architecture. Keras, and Scikit-learn libraries tools built on top of NumPy, SciPy, Matplotlib libraries for standard data mining processes such as data cleaning, pre-processing, normalization, visualisation and classification were implemented in Python.

The network topologies and payload information values of the NGP dataset containing 214,580 Modbus network packets with 60,048 packets associated with cyber attacks are used. These attacks contains 7 different attack categories with 35 different specific attack

types as explained in [252]. Each of these attacks represent one possible technique used in one of the APT steps. These attack categories also align with APT lifecycle. They represent different types of possible attacks that can occur at different level of ICS network layer described in Sub-section [2.3.4.1] - Purdue Model for ICS architecture. The number of records in each of these categories and the main four types of attacks as contained in the NGP data are shown in Figures [3.6] & [3.7]. The batch size of 64 & 124, epoch between 300 to 500 are run with a learning rate set in the range of 0.01-0.5. All the 17 features of NGP data and 49 features of UNSW-NB15 data are used as input vector with 70% as training set and 30% as validation set for the multi-attack classification. The training dataset were normalized between 0 to 1. The UNSW-NB15 data has a separate data for training and testing set.

The simulated APT_alert_dataset containing eight detectable APT attack techniques were also used to evaluate the capability of APT_{DASAC} to detect each of these individual techniques used during APT campaign as contained in this data (see Table [4.2]). These simulated detectable attack techniques are *disguised_exe_alert, hash_alert, domain_alert, ip_alert, ssl_alert, domain_flux_alert, scan_alert, tor_alert.*

In adition to the main approach implemented, traditional ML algorithm such as *Decision Tree (DT)* was also explored. Result from this was compared to the result achieved from implementing *"stacked Deep ensemble RNN-LSTM variants"* in order to further evaluate the APT steps detection capability of implementing the proposed framework.

5.2 Hyperparameters Settings Used

As mentioned in Sub-section [4.3.2] of Chapter [4], that *"the rate of attack detection is affected by the parameters used as these parameters have direct impact on attack detection. Based on this, several experiments with different network configuration were implemented to find the best optimal values for parameters such as number of epochs, batch_sizes, dropout, learning rate, network structure etc".*

Proposition 3: Hyperparameters Settings Used

- ☐ Batch_sizes: 64, 128 & 256
- ☐ Train_Test data_size: 67%_33%
- ☐ Learning rate: 0.01 to 0.5 with polynomial decay over all the epochs.
- ☐ Epochs: 100, 300 & 500 epochs.
- ☐ Neural Network Layer: Four layers was used (RNN-LSTM)
- ☐ Activation Function:
  - Each of the hidden layers has a ***sigmoid or ReLU*** activation function applied to produce non-linearity. This transforms the input into values usable by the output layer.
  - The ***softmax*** function is applied to the output layer to get probabilities of categories. This also helps in learning with ***cross-entropy loss*** function.
- ☐ Loss Function: ***categorical cross-entropy*** is applied as *loss function* to calculate the error rate.
- ☐ Optimizer: Adaptive Moment Estimation ***(Adam)*** optimizer is used for the back propagation to minimise the loss of *categorical-cross entropy.*

- ☐ Regularization: The ***dropout*** is used to alleviate the over-fitting (used as regularization technique to prevent over-fitting in neural networks. This randomly removes the units along with connections.

Considering the impact of a model hyperparameter setting, the experimental study focused on two tasks; the first task was focused on deriving hyperparameter values for best model performance. The second task applied the derived hyperparameter values so as to measure the APT_{DASAC} performance. The *Sigmoid & RELU activation function* was used through time with *ADAM* as optimizer, and the *categorical cross-entropy* is used as *loss function* to calculate the error rate.

5.3 Implementation of the Detection Framework

The implementation framework is carried out in three stages as explained in Chapter [4]; (i) Data input and probing stage - which involves data gathering and processing phase, (ii) Analysis & correlation stage and final phase, (iii) Decision phase, which is the detection and prediction phase. This proposed framework took advantage of supervised ML approach for detection and prediction.

5.4 Detection Module

The detection module uses stacked ensemble RNN variants to achieve its functionality. The process of preparing the used dataset, training the prediction model and the actual attack type detection using the model are all implemented in Python.

5.4.1 Preparing the Used Dataset

Deployed processes for preparing dataset for ML usage can have a great impact on system classification, detection and the quality of the developed model detection performance. However it is important to understand the type of data and/or network traffic flow of the domain of application.

This study, developed a dedicated module for data processing called *process_data*, imported as *pro* using Algorithm [5]. This *process_data* module applies different phases (discussed in Chapter [4], Sub-section [4.3.1]) based on data input type to obtain a transformed data (see Figure [5.1]), in a valuable format ready to be used to train the model for onward classification, detection and prediction processes.

These phases are (see: Code Snippet [C.2-C.5]), as used to accomplish these individual modules.

- ☐ data collection
- ☐ data cleaning
  - * feature transformation;
  - * dealing with missing data if present
  - * dealing with categorical data
  - * data balancing (optional)
  - * handling imbalance data (optional)
- ☐ data scaling & normalization

The *first phase* involves information gathering from the designated source. In the *second phase - data cleaning*; majority of ML classifiers work well with numeric values, during the data transformation of the cleaning phase, any columns which are not numeric are converted into numerical value format for the ML dataset. Any missing data for whatever reason(s) are also managed.

```
>>> X
array([[-0.00214307,  0.04024802,  0.25025262, ..., -0.13935674,
         0.15189011, -0.09448389],
       [-0.0023542 ,  0.04421311,  0.27490658, ...,  0.14408026,
         0.16685377,  0.14192545],
       [-0.0066672 , -0.22383485, -0.38991228, ..., -0.38930617,
         0.47253737, -0.09403773],
       ...,
       [-0.00523998, -0.17591948, -0.30644542, ..., -0.30596906,
         0.37138332, -0.62271152],
       [-0.00686302, -0.2304089 ,  0.08624861, ...,  0.28501747,
        -0.4867226 , -0.42717773],
       [-0.0065179 ,  0.12240949, -0.38118063, ...,  0.44181352,
        -0.46224676,  0.18712488]])
```

FIGURE 5.1: Transformed Data

Also, oversampling techniques such as SMOTE that involves process of oversampling instances from the minority class thereby creating new synthetic instances of that minority class were used to manage data imbalance. The use of SMOTE preprocessing algorithm is been considered as a "de facto" standard in the framework of learning from imbalanced data as mentioned in [282]. The *set().union() function* is deployed to balance the training and testing datasets.

However, in the *third phase - data scaling & normalization*; it is usually very common to have a dataset that its' features magnitudes of the units and range varies. Thus, the developed model tends to lean toward parameters with higher weight. In other to avoid any bias of model leaning toward parameters with higher weight, data parameters are scaled down between the range of 0 & 1 using Zscore normalization techniques. This is to calculate each observation in the dataset for the feature.

Algorithm 5: Data Pre-processing

```
- Pseudocode for Data Pre-processing
Begin
    Get Raw Dataset                                   /* gather raw data */
    while Get data do
        Step 1: Input the sample dataset
        Step 2: Convert the symbolic attributes features
        Step 3: return new set of data
        Step 4: Separate the instances of dataset into classes (y)
                (such as
                  Normal, Naïve Malicious Response Injection (NMRI)
                  Complex Malicious Response Injection (CMRI)
                  Malicious State Command Injection (MSCI)
                  Malicious Parameter Command Injection (MPCI)
                  Malicious Function Code Injection (MFCI)
                  Denial of Service (DoS), and
                  Reconnaissance (Recon))
                                  /* encode classes text values to indexes */
        Input: Categorical data
        Output: Classes indexes
        Data: Classes text value
        Def encode_text_index(df, name):
            le = preprocessing.LabelEncoder()
            df[name] = le.fit_transform(df[name])
            return le.classes                         /* classes indexes */

    end
    Step 5: Scale & Normalize data (x_t) into values from
            [0 − 1] using Equation [4.1]
                                  /* encode a numeric column as Zscores */
    Input: Training & Testing Datasets
    Output: Mean & Standard deviation (sd)
    Data: train_data & test_data
    Def encode_numeric_zscore(df, name, mean = None, sd = None):
        if mean is None: then
            mean = df[name].mean()
        else
            if sd is None: then
                sd = df[name].std()
            end
        end
        df[name] = (df[name] - mean) / sd
        return df, mean, sd
    Step 6: Split dataset into training and testing data
```

```
/* Data Pre-processing continued */
36
                    /* balance & reshape training & testing dataset */
37  while Step 7: Balance & Reshape Training & Testing sets do
38      Step 7a: Balance & reshape the training & testing
39          data features using Code Snippet [C.3] (Balancing
40          Data Features)
41  end
    Input: Training & Testing Datasets
    Output: Balanced & Reshaped Datasets
    Data: train_data & test_data
42  Def balance_df(train_data, test_data):
43      train_data_columns = list(train_data)
44      test_data_columns = list(test_data)
45      column_union = list(set().union(train_data, test_data))
46  for i in column_union: do
47      if i not in list(train_data): then
48          train_data[i] = 0
49          reshape train_data[i]                    /* reshape train_data */
50
51      else
52          if i not in list(test_data): then
53              test_data[i] = 0
54              reshape test_data[i]                  /* reshape test_data */
55
56          end
57      end
58  end
59  return balanced & reshaped train_data, test_data
60      Step 7b: return transformed new balanced & reshaped training
61          & testing datasets
                                                      /* pickle train_data */
62      Step 7c: Pickle transformed training data into a byte
63          stream and store it in a file/database (.pki)
64  for i in range(n): do
65      Pickle train_data into a byte [x]
66      while i < n: do
67          repeat
68              Pickle train_data into a byte [x]
69          until there is no sample left in the training set
70      end
71  end
72  return (classes instances indexes)
73  return (processed test_ data)
74  return (processed pickled train_ data)
75 end
```

5.4.2 Implementation of Alert Filtering, Clustering and Correlation Modules

Algorithm [5]-[8] are utilized to demonstrate the implementation of the proposed framework. The processed network traffic data cluster is recorded into a specific dataset in a format that is usable for the prediction module. Each data cluster contains alert step that are detectable. Algorithm [5] is utilized for data pre-processing, Algorithm [6] is utilized for alert filter module to filer each *generated_new_alert*, while Algorithms [7] & [8] are used for data clustering and steps detection respectively.

5.4.2.1 Implementation of Alert Filtering Module (AFM)

The individual detectable APT_{DASAC} attack defence points as discussed in Sub-section [4.3.3] generates attack alert. Each generated attack alert is filtered and written to the correct cluster table. The Algorithm [6] is used to filter alert into appropriate *alert_cluster_table*. Prior to writing the *generated_new_alert* into the alert cluster table, the alert filter module (AFM) checks if the same alert has been generated within the last given *time_window* (correlation window) to determine if the alert already exist within the *alert_cluster_table*. During the check process, if the alert already exist in the cluster table with the same attributes, the alert is disregarded, else if it doesn't exist, the new generated alert is written into the cluster table, ready to be used in the next stage, which is attack step correlation.

Algorithm 6: Alert Filtering Module (AFM)

```
- Pseudocode for Alert Filtering Module (AFM);
Given the time_window (tw);
begin
    Get the time_window;
    Get the alert_cluster_table;
    Get the generated_new_alert from any of the attack defence points;
    if the generated_new_alert exist in the alert_cluster_table then
        Disregard the generated_new_alert;
    else
        Write the generated_new_alert into the alert_cluster_table;
        Write the aligned alert_cluster_index to the cluster_data_group for ACCM;
    end
end
```

5.4.2.2 Implementation of Alert Clustering & Correlation Module (ACCM)

The ACCM module is used to determine the APT steps the new triggered alert belongs to. Four detectable attack step is contained in the cluster table, based on these four detectable alert steps, the ACCM will produce four clusters representing four different detectable attack steps of APT lifecycle. The first step is not detectable, the second step of APT lifecycle is the first detectable step denoted as *S1_alert* (see Sub-section [3.5.1] for the rest of the four detectable alert steps group). The ACCM starts a new cluster and writes the new alert into *S1_alert* cluster once step two of APT lifecycle is triggered.

Given a *time_window (tw)*, based on the number of filtered alert on the *alert_cluster_table* generated by AFM, the processing decision engine is designed to process each of the alert data to determine the type of attack step that the alert belongs to within APT lifecycle. The

data clustering is processed in such a way that if the first alert is detected, which is the second step of APT lifecycle, it is written into a table as cluster *S1_alert* and assign to cluster data *group_1*, while the process continues. If a new attack step is detected or triggered, and satisfy these conditions; i) the new triggered alert is same alert type, ii) and has the same attributes as those in *group_1*, the new alert is filtered out and ignored.

However, if the new triggered attack step is detected, and it does certify these conditions; (i) alert is the same type, (ii) but occurred outside the given correlation time window; (a) an alert email with regards to the triggered malicious alert is sent to the network security team for further forensics investigation, followed by appropriate response, (b) the processing engine will open up and start a new cluster, and then assign to cluster data *group_2*, until all the data are been processed and placed in a cluster data group depending on the attack step condition fulfilled. Each individual alert represents a single detectable step of APT lifecycle, and each cluster group will contain all the four detectable steps to be declared a full APT scenario. The Algorithm [7] is used to implement this module, ACCM.

Algorithm 7: Detection Module Implementation - Alert Generation

```
- Pseudocode for Detection Module Implementation
Begin
    Get Processed Dataset                          /* fetch processed_data */
    while Get data do
        Step 1: input the sample processed dataset
        Step 2: create classes indexes based on data features
        Step 3: return classes indexes
            /* correlate individual classes to cluster group indexes
           */
        Input: processed_data
        Output: generated_new_alert
           /* 1_NMRI_alert, 2_CMRI_alert, 3_MSCI_alert, 4_MPCI_alert,
           5_MFCI_alert, 6_DoS_alert, & 7_Recon_alert */
        Def correlate_new_alert(generated_new_alert):
        for i in range(n): do
                                                   /* generate new alert */
            generate_new_alert for class_type [x]
            while i < n: do
                Get cluster_module  /* individual detectable attack step
                 module */
                check the cluster_index to add the generated_new_alert
                check if the generated alert belong to any class index:
                 correlate_new_alert(generated_new_alert):
                    if the generated_new_alert belongs to cluster_index of
                     cluster_module : then
                        write the generated_new_alert to the cluster_module
                    else
                        if the generated_new_alert exist in cluster_module: then
                            suppress generated_new_alert
                        end
                        While Get data repeat
                            generate_new_alert for class type [x]
                            check
                            generated_new_alert against cluster_module
                            suppress
                            existing generated_new_alert
                            or write
                            generated_new_alert to cluster_new_alert
                        until there is no processed sample dataset left
                    end
            end
            return (generated_new_alert_index)
            return (cluster_module_index)
            return (correlated_new_alert data)
        end
    end
```

Algorithm 8: Detection Module Implementation - Generated Alert Correlation

```
1  - Pseudocode for Detection Module Implementation continued
2  Begin
3      Get clustered_alert_data from clustered_data
4      while Get clustered_alert_data do
                                              /* determine attack_class_type */
       Input: clustered_alert_data
       Output: attack_class_type
5      Def attack_class_type(clustered_alert_data):
6      for i in range(n): do
                                                  /* get_alert_cluster_index */
7          while i < n: do
8              if the generated_new_alert == NMRI_1_alert Or
                generated_new_alert == CMRI_2_alert: then
9                  call the decision_processing_engine :
10             else
11                 if the generated_new_alert == MSCI_3_alert Or
                    generated_new_alert == MPCI_4_alert Or
                    generated_new_alert == MFCI_5_alert: then
12                     call the decision_processing_engine :
13                 else
14                     if the generated_new_alert == DoS_1_alert Or
                        Recon_7_alert: then
15                         call the decision_processing_engine :
16                     else
17                     end
18                 end
19             end
20             decision_processing_engine              /* decision engine */
21             for each correlated cluster from clustered dataset do
22                 compile_campaign_cluster[x]
23                 if clustered_alert == campaign_alert_scenario And within
                    mapped_window_time then
24                     Write campaign_alert into the matching campaign_alert
                        scenario
25                     goto the next clustered_alert_data
26                 else
27                 end
28                 While Get clustered_alert_data repeat
29                     decision_processing_engine process
30                 until there is no clustered_alert_data sample left
31             end
32             return (attack_class_type)
33             return (attack_campaign)
34         end
35     end
36 end
```

5.4.3 Training the Detection Model

The training procedure model uses the pre-processed data from Sub-section [5.4.1] and Algorithm [5] to create train & test dataset, compile and fit the model. The parameters and hyperparameters settings as itemized in Proposition [5.2] is used to configure the training module. Each of these parameters in the dataset represents a sensor time series data which will be used at a pre-determined time window w of each iteration until no data is remaining.

The RNN variants and decision tree are used to train the model. The prediction accuracy of each trained model is calculated and the best model with highest prediction accuracy is saved to be used for model overall detection performance.

5.4.4 Applying the Ensemble Module for Detection

This module uses the probability function (created specifically for this module) for multi-classification to classify individual

- ☐ class/attack stages
- ☐ predicts
- ☐ calculate accuracy
- ☐ calculate overall performance accuracy

If a new threat step or alert is identified, it is correlated to data with the same cluster attribute under the same correlation index number which has been previously identified. Then, the trained individual prediction model is applied and the realized output is saved. It uses the newly correlated data attributes (best saved model) which has been trained and saved as the best predicted model. The probability function is invoked at this point, which then loads the predicted output best model (.hdf5) of each individual algorithm to determined the probability confidence of the average prediction accuracy of best saved model and then apply the ensemble function to generate the overall performance accuracy and the f1-score of the model.

This sub-section demonstrate the overall procedure of model outcome prediction phase. It is important to note that this phase of the framework is designed to predict trends of individual attack steps and calculate the overall classification of the predicted values accuracy. When all the steps are individually predicted, then the overall correlated predicted output is fed into the classification phase to classify and label the attack steps for individual model performance.

At this point, the network security team can use the achieved result to determine the probability of these individual predicted step to complete the APT lifecycle, and then apply all the necessary required procedure to stop and mitigate the attack before completion, thereby terminating their actions towards achieving the attackers final goal.

5.5 Summary

The implementation of APT_{DASAC} framework has been discussed in this chapter. The stacked deep ensemble RNNs variants approach is utilized in the implementation of the detection modules on the new gas pipeline simulated transactions between a RTU and MTU within SCADA-based gas pipeline containing *network topologies* and the *payload information* values of SCADA systems. These network topologies and payload information are very important to understand functionality of SCADA system, it's performance and to

detect if the system is in an out-of-bounds or critical state [1]. This approach was also deployed on two other enterprise datasets, the UNSW-NB15 dataset created by Australian Centre for Cyber Security [2] and the KDDCup'99[3] dataset to further evaluate and validate the performance of implementing APT_{DASAC} in different domains. Different ML approach such as decision tree were also explored.

Lastly, APT_alerts_dataset.db[4] - a historic record of APT alert generated through a monitored network, and it's associated information such as the malicious connection which includes seven features (*alert_type, timestamp, src_ip, dest_ip, src_port, dest_port, infected_host, malicious_item*) written into a log for each particular APT TTPs detection was utilized. This data was used to demonstrate the capability of the proposed framework to detect each detectable step of APT attack step. All the implementation and detection module are carried out using Python language and Jupyter notebook as interactive platform to achieve its functionality, performance, and outputs presented on this thesis.

[1] Modbus: https://simplymodbus.ca/TCP.htm Accessed on 10/03/2021
[2] ACCS: https://www.unsw.adfa.edu.au/unsw-canberra-cyber
[3] KDDCup'99: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[4] APT_alert: https://repository.lboro.ac.uk/articles/dataset/Dataset_of_Advanced_Persistent_Threat_APT_alerts/7577750

Chapter 6

Evaluation Results

***

6.1 Evaluation Metrics . . . 107
6.2 Experimental Evaluation of APT_{DASAC} Setup . . . 109
6.3 Evaluation of Application Domains . . . 109
6.4 Performance Evaluation of APT_{DASAC} . . . 120
6.5 Results and Discussion . . . 121
6.6 Mitigation of APTs at different Stages in the Lifecycle . . . 145
6.7 Summary . . . 147

In this chapter, the evaluation of APT_{DASAC} is introduced and the achieved results are discussed. The evaluation metrics used were also mentioned. A comparison of this APT_{DASAC} developed system and it's performance to implemented ML in four different domains and other existing proposed systems were also provided.

6.1 Evaluation Metrics

Conventionally, accuracy is usually used as a traditional way of classification performance measure. As mentioned by Yanmin et. al in [279]. Using accuracy as a metric measure is not appropriate when dealing with multi-class events data since the minority class may have little or no contribution when compared to majority classes toward achieved accuracy.

Proposition 4: Evaluation Metrics Used

For this reason, precision, recall, f1-score, overall accuracy, area under the curve (AUC) receiver operating characteristic (ROC) and confusion matrix are utilized to validate the approach of using RNN variants for APT detection system *"APT_{DASAC}"* proposed in this thesis to get a clearer understanding of the output. All the metrics calculation are based on true positive (TP), true negative (TN), false positive (FP), false negative (FN), *Loss* and *confusion Matrix*.

1. TP - abnormal instances correctly predicted as abnormal.
2. TN - normal instances correctly predicted as normal
3. FP - normal instances incorrectly predicted as abnormal

4. FN - abnormal instances incorrectly predicted as normal

5. *Confusion Matrix* - is a useful tool, good and simple matrix used in this thesis to get a general overview of model performance (see: Code Snipet [C.6]).

TABLE 6.1: The Confusion Matrix Description for Multiple Classes: with letter A–E representing each class of the detectable classes as contained within the APT_alerts_dataset.db, NGP_5 and KDDcup'99 datasets used. Additional columns and rows are required to generate confusion matrix for NGP_6 and UNSW-NB15 dataset (not shown on this table).

		Predicted				
		A	**B**	**C**	**D**	**E**
Actual	**A**	TP_A	E_{AB}	E_{AC}	E_{AD}	E_{AE}
	B	E_{BA}	TP_B	E_{BC}	E_{BD}	E_{BE}
	C	E_{CA}	E_{CB}	TP_C	E_{CD}	E_{CE}
	D	E_{DA}	E_{DB}	E_{DC}	TP_D	E_{DE}
	E	E_{EA}	E_{EB}	E_{EC}	E_{ED}	TP_E

6. *Precision (P)* - is the ability of a classification model to identify only the relevant data points, that is the ratio of TP records over the sum of TP & FP. P can be calculated using Equation (6.1).

$$P = \frac{TP}{(TP + FP)} \tag{6.1}$$

7. *Recall (R)* – is the ability of a model to find all the relevant cases within a given data, that is the ratio of the TP records over the sum of TP & FN, as illustrated with Equation (6.2). Recall is also referred to as probability of detection, true positive rate (TPR) or sensitivity (S).

$$R = \frac{TP}{(TP + FN)} \tag{6.2}$$

8. *f1-score (f1)* - is referred to as the weighted average of precision and recall, that is the harmonic mean of precision and recall of a class in one given metric. $f1$ is represented as Equation (6.3), the weight parameter is denoted by β^2 and is usually set to 1 by default, it measures the trade-off between recall and precision.

$$f1 = \frac{(1 + \beta^2).P.R}{\beta^2 P + R} \tag{6.3}$$

9. *AUC-ROC curve* - the *precision-recall* curve shows the trade-off between P & R at different threshold settings, where *AUC* measures the degree of separability and *ROC* represent the probability curve. It is also a useful technique for visualising model output, organising and selecting classifiers based on their performance as described in [299]. A high *AUC* represents both high R and high P, where high P relates to a low *false positive rate (FPR)*, and high R relates to a low *false negative rate (FNR)*. High scores for both P & R show that the model is returning accurate results P, as well as returning a majority of all positive results R. An ideal model with high P and high R will return many results, with all results labelled correctly. The *AUC-ROC* curve can

also be used as a scalar measure rather than the higher the *AUC* value, the better the model as highlighted in [276].

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2} \tag{6.4}$$

10. *Overall Accuracy (OaAcc)* - measures the rate of the correctly classified class instances of all the classes (attacks and normal). An overall classification performance is an important performance matrix require to evaluate the overall model performance rate, it can be calculated as represented in Equation (6.5).

$$OaAcc = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{6.5}$$

6.2 Experimental Evaluation of APT_{DASAC} Setup

To evaluate the performance of the implemented model "APT_{DASAC}", Jupyter Notebooks, Python language, Keras, and Scikit-learn libraries tools built on top of NumPy, SciPy, Matplotlib libraries and pandas for data processing & result display were used. The model application focused on identifying different techniques used at different stages of APT lifecycle during APT attack scenario to exploit and attack their target.

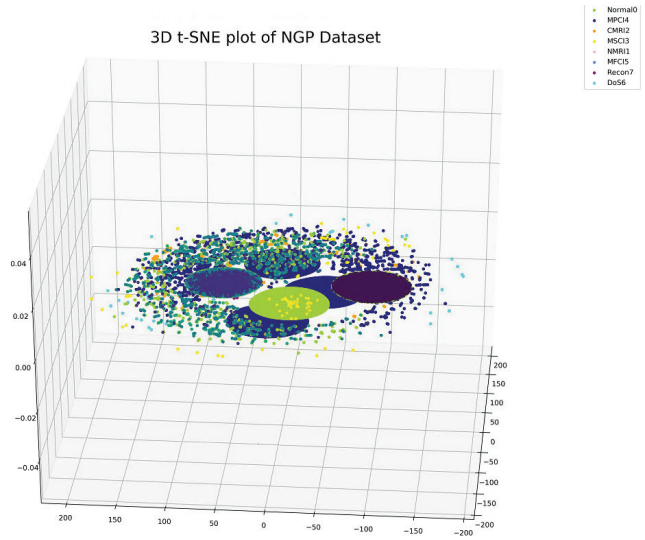
Due to the imbalance nature of the implementation data, which reflected to the nature of APT steps distribution scenario, the implemented model interest was on reducing the *Type I error (FPR)* & *Type II error (FNR)* of each individual attack step as to achieve the highest posible detection rate for that particular attack class. Thus, both *Precision* and *Recall* metrics measure are considered to derive the *f1-score* for the implemented model performance.

Precision metric which is also referred to as positive (+ve) prediction value *(PPV)* is employed to determine the actual predicted positive values out of all the total predicted positive result. Example: given step *"3"*, how correct does it predict step "3". *Recall* metric, also referred to as *TPR* or *sensitivity* is applied to calculate, out of the total predicted actual values, how often or how many values were correctly predicted positively. Example: given a rare attack step *"3"*, how often does the model predict the given step "3" ? Hence, *f1-score* is considered as all the implementation datasets are imbalance data in nature, thus the reason for considering f1-score to find the harmonic mean of both *Precision* and *Recall* .

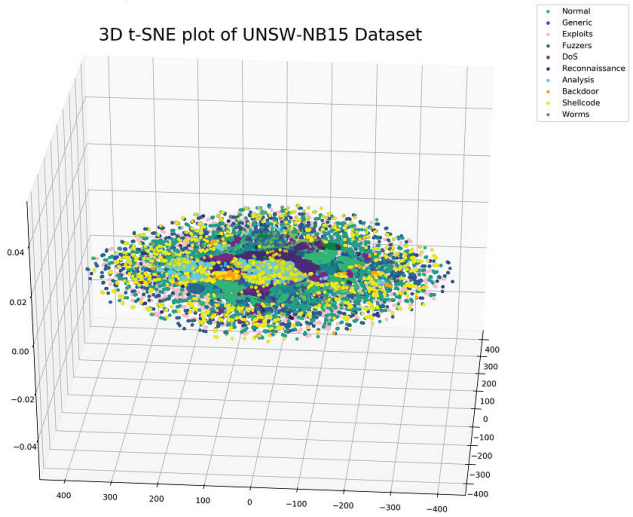
6.3 Evaluation of Application Domains

Four different case studies as implemented will be discussed where the framework has been used to detect attack steps. The first case study involves the use of the framework in the detection of cyber-attack data injection attack activities in SCADA gas pipeline Systems. The second study is on KDDCup'99 data that contains a wide variety of intrusions activities simulated within military network environment. The third scenario used UNSW-NB15 data containing a hybrid of real modern normal and contemporary synthesized attack activities of the network traffic. In the fourth experiment, APT_alerts_dataset.db, a simulated data which contains a historical record of APT attack steps, built from a monitored network over six months period is utilized.

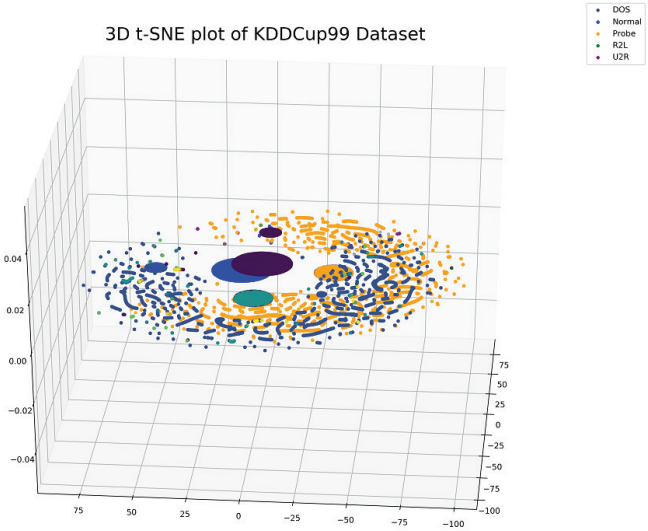
The detailed description and sources of all the used datasets are given in Section [3.5]. The t-SNE (t-distributed Stochastic neighbor embedding) 3D data visualization of each of the first three datasets records' distribution are shown in Figure [6.1]. All features as contained within each case data were used as input vector with 67% as train set and 33% as test set, ensuring an even distribution of alerts steps for the four cases.



(A) NGP Dataset



(B) UNSW-NB15 Dataset



(C) KDDCup'99 Dataset

FIGURE 6.1: t-SNE 3D Visualization Projection of NGP, UNSW-NB15 and KDDCup'99 Datasets Records Distribution

The t-SNE techniques is employed to visualize the similarity data that is capable of retaining the local data structure, and to understand some importance of the used experimental dataset's global structure in high-dimensional data plane, thereby projecting it into low-dimensional 3D space. This is achieved by giving each data point a location in a three-dimensional map as the case of this work. This technique for data visualization is applied as it helps to alleviate both the crowding data point problem for multiple classes and the optimization problems of SNE (Stochastic Neighbor Embedding [300]) as illustrated in [301]. This makes it extremely useful for this thesis as it is a nonlinear dimensionality reduction tool, although it was not used as a tool to perform dimensionality

reduction for the proposed model training set.

A closer observation of Figure [6.1] produced by applying the t-SNE technique, reveals the natural classes in each of the dataset by representing the one-dimensional manifold of viewpoints as a closed loop. The t-SNE distorts the loop of objects which look similar from the front and the back by mapping nearby points. Some of this produced cluster may contain some data points that are clustered with the wrong class. Although most of these points correspond to distorted points for which many are difficult to identify (data points for rare attack event). This highlights the issue of correctly detecting and classifying classes with fewer records for minority classes. The visualization of high-dimensional data is an important problem in many different domains, and deals with data of widely varying dimensionality [301].

6.3.1 Case Study One Evaluation – Application to The New Gas Pipeline Control System Dataset

In the first experiment, APT_{DASAC} was applied to a simulated data containing attacks and normal network behaviour from a laboratory scale industrial control system; a gas pipeline control system in form of *.arff file*. This data contains network transaction captured over serial line that contains network information (e.g., time stamp, station address, etc), payload information (e.g., system control and state information) and label (e.g., binary, categories, and specific mode identifier).

As part of this study in [52], the focus was on command injection (CI) attack (attack type that alters the system behaviour through injection of false control and configuration commands into a control system) and response injection (RI) attacks (this type of attack modifies the response from server to client, thereby providing false information about system state). A multi-stage approach based on DL techniques was applied. This approach is validated with two case scenarios; a network transactions between a RTU & MTU in-house SCADA gas pipeline control system and a case study of command and response injection attacks detection. This implemented approach achieved competitive attack detection capability with 0% FAR and TPR of 96.50%.

This was investigated further in [5], where stacked ensemble-LSTM variants for APT_{DASAC} framework were applied. This approach combines networks' results as to optimize attack detection rate and achieved overall average mean detection accuracy and validation average accuracy of 85%.

The performance classification report on NGP data are shown in Figures [6.2] and [6.3]. Where Figure [6.2], is the result of experiment that focused on the main four detectable types of attack as contained within the data which aligns to four detectable steps of APT lifecycle. Figure [6.3], is the report derived when the categorical attacks as contained in NGP dataset is considered as APT steps, grouped into five steps of APT to align with APT lifestyle as explained in Figure [2.1.3].

Classification Report for APTDASAC-LSTM Result

	precision	recall	f1-score	support
Command_Injection	0.96	0.53	0.68	10959
DoS	1.00	0.45	0.62	718
Normal	0.85	1.00	0.92	70812
Reconnaissance	1.00	0.87	0.93	1279
Response_Injection	1.00	0.02	0.03	6860
accuracy			0.86	90628
macro avg	0.96	0.57	0.64	90628
weighted avg	0.88	0.86	0.82	90628

(A)

Classification Report for APTDASAC Result

	precision	recall	f1-score	support
Command_Injection	0.96	0.52	0.68	10959
DoS	0.99	0.44	0.61	718
Normal	0.85	1.00	0.92	70812
Reconnaissance	1.00	0.87	0.93	1279
Response_Injection	1.00	0.02	0.03	6860
accuracy			0.86	90628
macro avg	0.96	0.57	0.63	90628
weighted avg	0.88	0.86	0.82	90628

(B)

Classification Report for ML-DT

	precision	recall	f1-score	support
Command_Injection	0.46	0.67	0.55	10959
DoS	0.42	0.45	0.43	718
Normal	0.88	0.67	0.76	70812
Reconnaissance	0.13	0.98	0.23	1279
Response_Injection	0.34	0.51	0.41	6860
accuracy			0.66	90628
macro avg	0.44	0.66	0.47	90628
weighted avg	0.77	0.66	0.70	90628

(C)

FIGURE 6.2: Classification report for all the algorithms on NGP_5 dataset

Classification Report for LSTM Result

	precision	recall	f1-score	support
Command_Injection	0.98	0.45	0.62	9343
DoS	1.00	0.43	0.60	718
Function_Code_Injection	0.94	1.00	0.97	1616
Normal	0.85	1.00	0.92	70812
Reconnaissance	1.00	0.90	0.95	1279
Response_Injection	1.00	0.01	0.03	6860
accuracy			0.86	90628
macro avg	0.96	0.63	0.68	90628
weighted avg	0.88	0.86	0.82	90628

(A)

Classification Report for APTDASAC Result

	precision	recall	f1-score	support
Command_Injection	0.97	0.44	0.61	9343
DoS	0.99	0.44	0.61	718
Function_Code_Injection	0.79	1.00	0.88	1616
Normal	0.85	1.00	0.92	70812
Reconnaissance	1.00	0.65	0.79	1279
Response_Injection	1.00	0.01	0.03	6860
accuracy			0.86	90628
macro avg	0.93	0.59	0.64	90628
weighted avg	0.88	0.86	0.81	90628

(B)

Classification Report for ML-DT

	precision	recall	f1-score	support
Command_Injection	0.98	0.95	0.96	9343
DoS	0.96	0.94	0.95	718
Function_Code_Injection	0.99	1.00	0.99	1616
Normal	0.96	0.97	0.97	70812
Reconnaissance	0.99	0.97	0.98	1279
Response_Injection	0.73	0.67	0.69	6860
accuracy			0.95	90628
macro avg	0.93	0.92	0.92	90628
weighted avg	0.95	0.95	0.95	90628

(C)

FIGURE 6.3: Classification report for all the algorithms on NGP_6 dataset

6.3.2 Case Study Two Evaluation – Application to KDDCup'99 Dataset

In the second experiment, KDDCup'99 data which was created with the intent of researchers to test viable IDS for effectiveness was used to establish the classification and detection capability of this model in identifying attacks and to correctly classify individual attacks to their attack step.

Although, this dataset has played a vital part in research community for evaluating computer network IDSPs and as a benchmark dataset for other researchers to compare and validate results, this data was found to contain unintended patterns which has led to algorithms to easily learn differences among patterns, making the detection rate very high as can be seen in this study classification report represented in Figure [6.4]. All the data features were used as input vector, trained on full set, and evaluated with the full test set.

Study in [51], applied an approach based on DL models which includes RNN variants algorithms and ML models on KDDCup'99 data. The achieved results indicates that DL based model seems to be more efficient when compared with result derived from ML based models as implemented on this study. However, the study did not consider the imbalanced distribution of rare attack activities, which may affect detection / classification rate. During the training, it was observed that the RNN variants seem to be more suitable when classifying high-frequency attacks and also the low frequency attacks with very low detection capability, achieving 62.50%, 56.20% and 37.50% for LSTM, GRU and RNN respectively on multi-attack step classification, while achieving a very high average accuracy of 99.99% for RNN variants on differentiating detectable attack steps from normal network activities (see Tables [3.7]-[3.8] of Chapter [2].

6.3.3 Case Study Three Evaluation – Application to UNSW-NB15 Dataset

The third experiment used The UNSW-NB15 dataset to establish the classification and detection capability of the proposed framework, *"APT_{DASAC} framework"* in identifying a single detectable attack step, able to correctly differentiate normal network & attack connection records, and categorize these attacks to their attack step. The achieved classification report of this case study is shown in Figure [6.5].

Study in [1], explored the applications of heterogeneous ensemble approach and SMOTE data resampling technique with focus on capturing the rare attack and achieved a maximum average mean accuracy of 81.02% with a significant validation loss for UNSW-NB15 data. Applying data resampling to provide a balanced data distribution, ranging from under-sampling over-sampling, or combination of both techniques to improve overall detection rate. Other examples of such techniques are one-versus-one approach (OVO) and one-versus-all approach (OVA) based on decomposition schemes [276].

Attackers utilizes multiple attack tactics to deliver an attack on their target. This multiple attack tactics leads to generation of uneven distribution of attack payload information among examples of different attacks. Learning from imbalance data distribution in multi-attack detection and multi steps classification problem, poses a significant challenge for detection algorithms based on ML, especially in detecting the rare attack steps. Studies based on ensemble supervised learning and problem decomposition with cost-sensitive learning are still an active research field in ML community as demonstrated by the authors [272], [273] and [274].

Classification Report for GRU Result

	precision	recall	f1-score	support
DOS	1.00	1.00	1.00	129181
Normal	1.00	1.00	1.00	32102
Probe	0.99	0.99	0.99	1355
R2L	0.97	0.94	0.95	372
U2R	0.92	0.71	0.80	17
accuracy			1.00	163027
macro avg	0.98	0.93	0.95	163027
weighted avg	1.00	1.00	1.00	163027

(A)

Classification Report for APTDASAC Result

	precision	recall	f1-score	support
DOS	1.00	1.00	1.00	129181
Normal	1.00	1.00	1.00	32102
Probe	1.00	0.99	0.99	1355
R2L	0.94	0.90	0.92	372
U2R	1.00	0.12	0.21	17
accuracy			1.00	163027
macro avg	0.99	0.80	0.82	163027
weighted avg	1.00	1.00	1.00	163027

(B)

Classification Report for ML-DT

	precision	recall	f1-score	support
DOS	1.00	1.00	1.00	129181
Normal	1.00	1.00	1.00	32102
Probe	0.98	0.98	0.98	1355
R2L	0.97	0.94	0.95	372
U2R	0.62	0.59	0.61	17
accuracy			1.00	163027
macro avg	0.91	0.90	0.91	163027
weighted avg	1.00	1.00	1.00	163027

(C)

FIGURE 6.4: Classification report for all the algorithms on KDDCup'99 dataset

Classification Report for RNN Result

	precision	recall	f1-score	support
Analysis	0.87	0.62	0.73	4122
DoS	0.56	0.05	0.09	4047
Exploits	0.60	0.94	0.74	11020
Fuzzers	0.70	0.72	0.71	6001
Generic	1.00	0.98	0.99	13200
Normal	0.94	0.90	0.92	18480
Worms	0.55	0.30	0.39	993
accuracy			0.82	57863
macro avg	0.75	0.64	0.65	57863
weighted avg	0.83	0.82	0.80	57863

(A)

Classification Report for APTDASAC Result

	precision	recall	f1-score	support
Analysis	0.88	0.62	0.73	4122
DoS	0.60	0.05	0.09	4047
Exploits	0.60	0.95	0.74	11020
Fuzzers	0.74	0.63	0.68	6001
Generic	1.00	0.98	0.99	13200
Normal	0.91	0.93	0.92	18480
Worms	0.58	0.26	0.36	993
accuracy			0.82	57863
macro avg	0.76	0.63	0.64	57863
weighted avg	0.83	0.82	0.80	57863

(B)

Classification Report for ML-DT

	precision	recall	f1-score	support
Analysis	0.32	0.79	0.45	4122
DoS	0.28	0.07	0.11	4047
Exploits	0.71	0.58	0.64	11020
Fuzzers	0.69	0.62	0.65	6001
Generic	0.99	0.98	0.99	13200
Normal	0.91	0.91	0.91	18480
Worms	0.40	0.22	0.28	993
accuracy			0.76	57863
macro avg	0.61	0.60	0.58	57863
weighted avg	0.77	0.76	0.75	57863

(C)

FIGURE 6.5: Classification report for all the algorithms on UNSW-NB15 dataset

6.3.4 Case Study Four Evaluation – Application to APT_alerts_dataset.db

In this fourth experiment, APT_{DASAC} was applied to a simulated record of malicious connection of data generated through a monitored network, then written into a log and stored as *.db file*. This data contains eleven different types of alerts of which eight alerts out of eleven are detectable alerts representing a single step of APT lifecycle. These eight alerts (*disguised_exe_alert, hash_alert, domain_alert, ip_alert, ssl_alert, domain_flux_alert, scan_alert, tor_alert*), belong to one of the four detectable steps of APT lifecycle, which are; *step 2, step 3, step 5* & *step 6*, mentioned in Sub-section [3.5.1]. The APT_alerts_dataset.db data was used to demonstrate the capability of the designed framework to detect each detectable step of APT attack step, thus highlighting APT campaign.

This implemented approach achieved competitive attack detection capability of 96.90% TPR & 0.04% FPR. The approach achieved an overall average mean detection accuracy of 86.73%. Further investigation with ML-based approach applied achieved TPR of 87.80% and average overall accuracy of 68.00%. The detectable steps performance report shown in Figure [6.6], where Figure [6.6a], is the result derived from application of APT_{DASAC} approach to detect every single alert representing each step of detectable APT attack as contained within the data which aligns to the four detectable steps represented in Figure [6.7] of APT lifecycle. Figure [6.6b], is the report derived when ML-based approach were applied to further investigate it's detection capability with this approach applied.

Classification Report

	precision	recall	f1-score	support
Data_Discovery_S5	0.87	0.86	0.86	405
Data_Exfiltration_S6	0.81	0.80	0.80	597
Delivery_S2	0.95	0.95	0.95	1000
Exploitation_S3	0.82	0.83	0.82	915
accuracy			0.87	2917
macro avg	0.86	0.86	0.86	2917
weighted avg	0.87	0.87	0.87	2917

(A) Detectable steps report - APT_{DASAC}

Classification Report

	precision	recall	f1-score	support
Data_Discovery_S5	0.76	0.62	0.68	405
Data_Exfiltration_S6	0.00	0.00	0.00	597
Delivery_S2	0.84	1.00	0.91	1000
Exploitation_S3	0.53	0.81	0.64	915
accuracy			0.68	2917
macro avg	0.53	0.61	0.56	2917
weighted avg	0.56	0.68	0.61	2917

(B) Detectable steps report - ML-DT

FIGURE 6.6: Result report for APT_{DASAC} & ML-DT on APT_alert_dataset Records

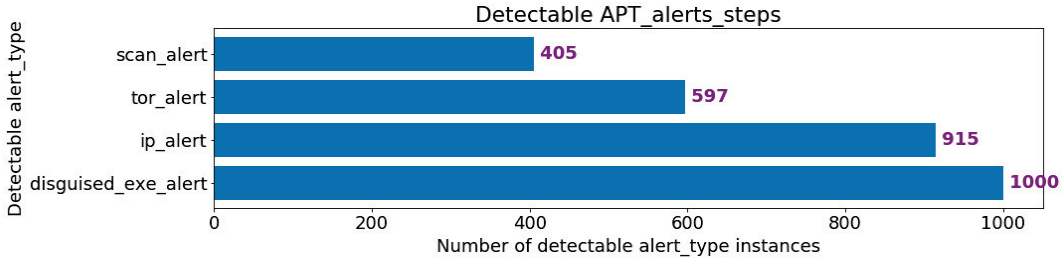


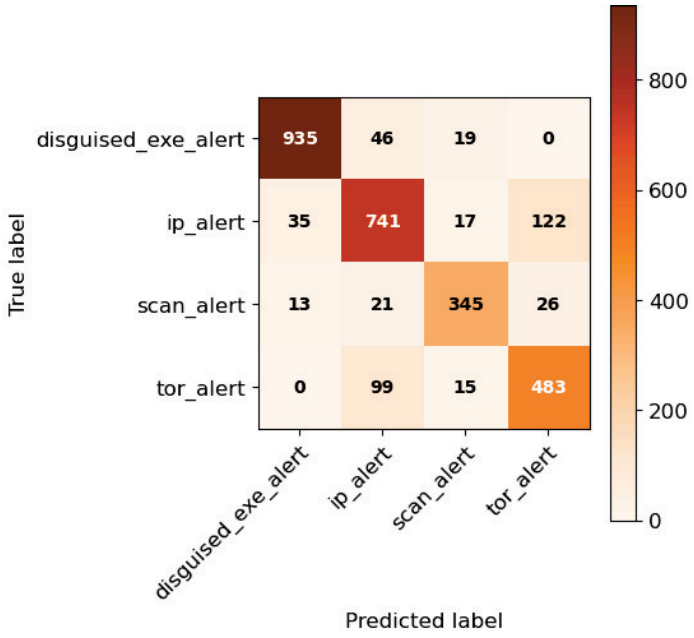
FIGURE 6.7: Four Detectable APT_ALert Steps

6.3.5 Individual Detectable Steps Evaluation Result

Figure [6.8], contains the output derived from detecting each individual detectable attack steps when APT_{DASAC} framework was implemented on APT_alerts_dataset.db. The confusion matrix shown in [6.8a,] shows the predicted and the actual true identified value of all the four detectable attack steps. This confusion matrix was used to calculate the TN, TP, FN, & FP numbers used to evaluate the *precision, recall, f1-score* and the *AUC_ROC curve* as shown in Figures [6.8b] & [6.8d] respectively. The achieved results are summarized in Table [6.2].

From the confusion matrix, we can see that the number of correctly classified *disguised_exe_alert*, belonging to detectable Step A of APT lifestyle (as in Table [4.2]) is 935 out of 1000 alerts, resulting to individual *Detection_rate* of 93.50% for Step A. *Precision & ROC curve* of 95.00%, with 94.00% for *Recall & f1-score* achieved. For *ip_alert* belongs to detectable Step B, 80.98% *Detection_rate* were achieved. Correctly predicted number of 741 out of 915 alerts, with *Precision & ROC curve*, of 82.00% & 86.00% respectively, 81.00% for *Recall & f1-score* respectively were achieved.

For *scan_alert* belongs to detectable Step C, 85.19% *Detection_rate* were achieved. Correctly predicted number of 345 out 405 alerts, with *Precision, ROC curve, Recall & f1-score* of 87.00% & 92.00%, 85.00% & 86.00% respectively were achieved. For *tor_alert* belongs to detectable Step D, 80.90% *Detection_rate* were achieved. Correctly predicted number of 485 out of 597 alerts, with *Precision, ROC curve, Recall & f1-score* of 77.00% & 87.00%, 81.00% & 79.00% respectively were achieved. Although, there are few missed point as can be seen on the confusion matrix. Computing the probability of detecting each APT step, the individual *Detection_rate* of these detectable steps (A, B, C,& D), of **93.50%, 80.98%, 85.19%, & 80.90%** achieved are commendable. The proposed model did very well, especially in detecting disguised executable file attack, thereby demonstrating the capability of the designed framework to detect each individual detectable step of APT attack step, thus highlighting APT campaign.



(A) Detectable steps Confusion Matrix

```
{'disguised_exe_alert': '93.50%',
 'ip_alert': '80.98%',
 'scan_alert': '85.19%',
 'tor_alert': '80.90%'}
```

(B) Individual detectable step detection

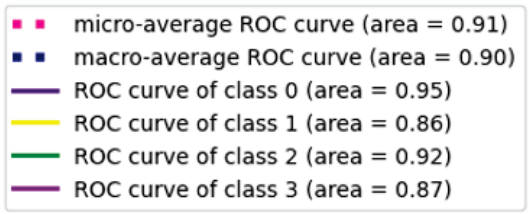
```
  Evaluation Report
======================================================
                    precision    recall  f1-score   support

disguised_exe_alert       0.95      0.94      0.94      1000
          ip_alert        0.82      0.81      0.81       915
       scan_alert         0.87      0.85      0.86       405
        tor_alert         0.77      0.81      0.79       597

          accuracy                            0.86      2917
         macro avg        0.85      0.85      0.85      2917
      weighted avg        0.86      0.86      0.86      2917

======================================================
```

(C) Detectable steps evaluation report



(D) Detectable steps AUC-ROC curve

FIGURE 6.8: Individual Detectable Steps Detection Result

TABLE 6.2: Summary of each individual detectable steps detection result of implementing APT_{DASAC} on APT_alerts_dataset.db evaluation. The mean average *Precision, Recall, f1-score, AUC_ROC curve* and overall *Detection_rate* of each individual detectable steps recorded against each step. The metrics parameter with best result for each individual step result are written in bold text.

Summary of Individual Detectable Steps Detection Result					
Detectable Steps	**Precision (%)**	**Recall (%)**	**f-1 score (%)**	**AUC_ROC(%)**	**Detection_rate(%)**
disguised_exe_alert (*StepA*)	95.00	94.00	94.00	95.00	93.50
ip_alert (*StepB*)	82.00	86.00	81.00	82.00	80.98
scan_alert (*StepC*)	87.00	85.00	86.00	92.00	85.19
tor_alert (*StepD*)	77.00	81.00	79.00	87.00	80.90

6.4 Performance Evaluation of APT_{DASAC}

The evaluation experiments were performed to test the developed APT_{DASAC} framework in terms of effectiveness and detection capabilities in four different domains. This study considered the potential vulnerabilities to data injection cyber attacks of gas pipeline, enterprise network, military environment network activities and a simulated APT steps alert data.

Developed an attack detection algorithm based on stacked ensemble-RNN variants approaches to evaluate the network transactions between RTU and MTU within SCADA-based gas pipeline and data communication between the system components information flow. These algorithm are used to detect and identify different types of attack representing attack steps such as interruption attack (DoS attack) used to block all communication between two system nodes. The capability of the framework to detect every single detectable step of APT attack steps mentioned in Sub-section [3.5.1], as to determine the possibility of a full APT scenario is also evaluated as described in Sub-section [6.3.4] (the fourth case study of the application domains).

The description of the proposed APT_{DASAC} framework is presented in Chapter [4]. The implementation approach including the implementation setup, the hyperparameters & parameter settings used, programming language, tools utilized and the implementation algorithms of each stages of the model are described in Chapter [5]. The model was trained on every dataset for each case and validated with every related dataset, recording the mean average Precision, Recall, f1-score, and overall average accuracy on Table [6.3]. The TPR, FPR, f1-score, & micro/macro-ave_roc curve recorded on Table [6.4] and macro f1-score, error, f1-score, and Overall average model accuracy on Table [6.5].

TABLE 6.3: Summary of the Average Result Report of all Algorithms on NGP_5, NGP_6, UNSW-NB15 and KDDCup'99 Datasets Evaluated. With mean average Precision, Recall, f1-score, and overall average accuracy recorded for different problem domains networks cross evaluation. The metrics parameter with best result for each individual algorithm are written in bold text.

Summary of Average Result Report on the Four Evaluated Datasets					
Dataset	**Algorithms**	**Precision (%)**	**Recall (%)**	**f-1 score (%)**	**Overall_average accuracy(%)**
NGP_5	RNN variants	88	86	82	86
	APT_{DASAC}	88	86	82	86
	ML-DT	77	66	70	66
NGP_6	RNN variants	88	86	82	86
	APT_{DASAC}	88	86	81	86
	ML-DT	95	95	95	95
UNSW-NB15	RNN variants	83	82	80	82
	APT_{DASAC}	83	82	80	82
	ML-DT	77	76	75	76
KDDCup'99	RNN variants	100	100	100	100
	APT_{DASAC}	100	100	100	100
	ML-DT	100	100	100	100
APT_alerts_dataset.db	APT_{DASAC}	87	87	87	87
	ML-DT	56	68	61	68

TABLE 6.4: Overall Summary Result of all Algorithms on NGP_5, NGP_6, UNSW-NB15 and KDDCup'99 Datasets. With TPR, FPR, f1-score, and micro/macro-ave_roc curve recorded for different problem domains networks cross evaluation. The metrics parameter with best result for each individual algorithm are written in bold text.

Summary Result of all Aglorithms on the four Evaluated Datasets					
Datasets	**Algorithm**	**TPR (%)**	**FPR (%)**	**f1-score (%)**	**micro / macro-ave_roc curve**
NGP_5	RNN variants	96.12	0.31	82.07	0.91 /0.72
	APT_{DASAC}	96.28	0.62	82.01	0.91 /0.72
	ML-DT	46.26	11.01	69.52	0.79 / 0.77
NGP_6	RNN variants	97.87	0.32	81.97	0.92 / 0.76
	APT_{DASAC}	97.41	0.00	81.45	0.92 / 0.76
	ML-DT	97.61	1.42	94.6	**0.97 / 0.94**
UNSW-NB15	RNN variants	86.99	1.75	79.8	0.89 / 0.80
	APT_{DASAC}	88.18	2.15	79.57	0.89 / 0.80
	ML-DT	31.84	7.05	75.02	0/86 / 0.78
KDDCup'99	RNN variants	99.98	0.08	99.93	0.93 / 1.00
	APT_{DASAC}	99.98	0.05	99.93	0.96 / 1.00
	ML-DT	99.99	0.05	99.92	0.95 / 1.00
APT_alerts_dataset.db	APT_{DASAC}	96.90	0.04	99.93	0.96 / 0.91
	ML-DT	87.80	3.38	99.92	0.82 / 0.80

6.5 Results and Discussion

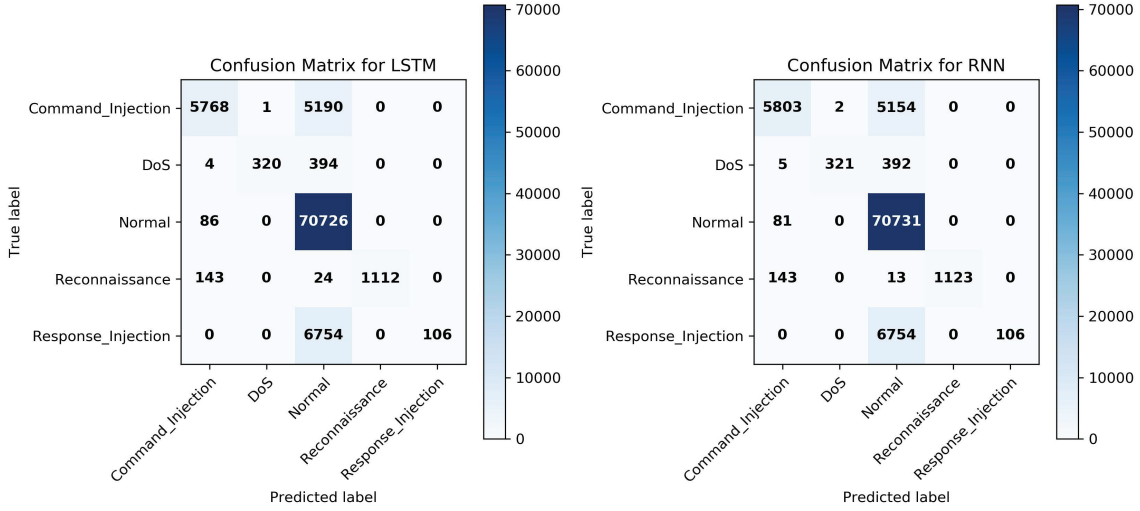
Each attack type within each dataset was split into train and test at 67% and 33%, ensuring an even distribution of each attack step labels for the four cases, then utilizing the hyperparameter settings as outlined in Section [5.2]. Each model is trained over a total of 300 epochs with exception of the fourth case scenario that was trained over 100 epochs. To validate this approach for detecting APT step attacks, the chosen statistical metrics highlighted in Section [6.1] are calculated to (i) evaluate the ability of this approach to accurately detect and identify an abnormal network as an attack step, (ii) check the ability of this model to detect different type of attack steps and correlate these attack steps to accurately (iii) get a clearer understanding of the output, and (iv) then deduce the actual overall performance of the model if compared to other existing approach in highlighting APT campaign.

TABLE 6.5: The macro f1-score, error, f1-score, and Overall average model accuracy recorded for the different problem domains networks cross evaluation on these different sets. The first two metrics parameter with best result for each metrics against each dataset are written in bold text.

Probability Result of the ensemble model on the four different dataset used				
Dataset	**macro f1-score (%)**	**Loss (%)**	**Val_Loss (%)**	**Overall accuracy (%)**
NPG_5	63.58	0.32	0.32	86.3
NPG_6	68.05	0.32	0.32	86.36
UNSW-NB15	65.42	0.42	0.42	82.19
KDDCup'99	94.08	0.01	0.01	99.92
APT_alerts_dataset.db	94.08	0.69	0.69	86.73

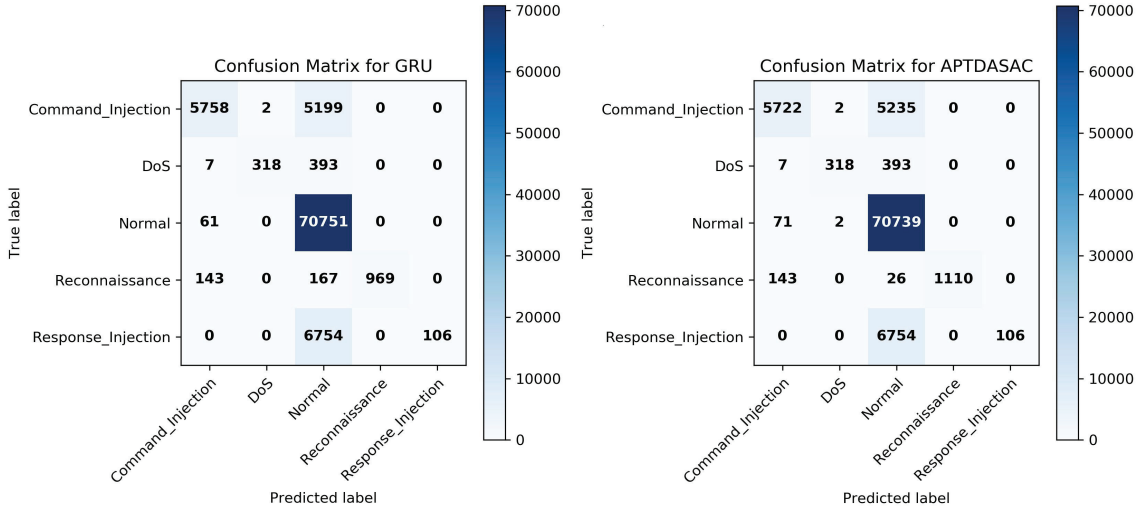
The report on Figures [6.2]-[6.6] gives the account of implementing APT_{DASAC} on the four case studies. The measured matrix values are also recorded in Tables [6.3]-[6.5]. As can be seen from the recorded results, all the implemented algorithms returned high average accuracy and detection rate for the four cases in identifying attack steps and classifying these attack step types. The average overall result from the proposed model result were also compared to ML-based model result in Sub-section [6.5.1].

Although, each algorithm returned a competitive average detection rate with insignificant *FPR* and validation loss, it was observed that most of the implemented algorithm appear to be suitable for identifying high-frequency attack steps with less detection capability for the low-frequency attack steps as it returned low detection rate. It was also observed that, each of these algorithm's result is slightly different in each case study. All the generated Confusion Matrices used for all the experimental reports calculation are shown in Figures [6.9], [6.12], [6.15], [6.18] and [6.21].



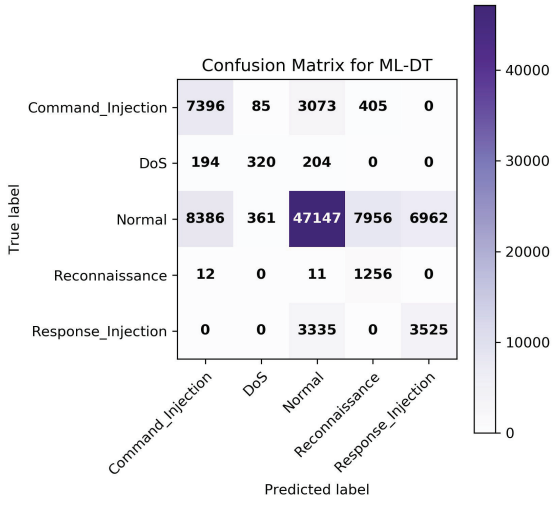
(A)

(B)



(C)

(D)



(E)

FIGURE 6.9: Four Main Attack Types and Normal Records for NGP Data

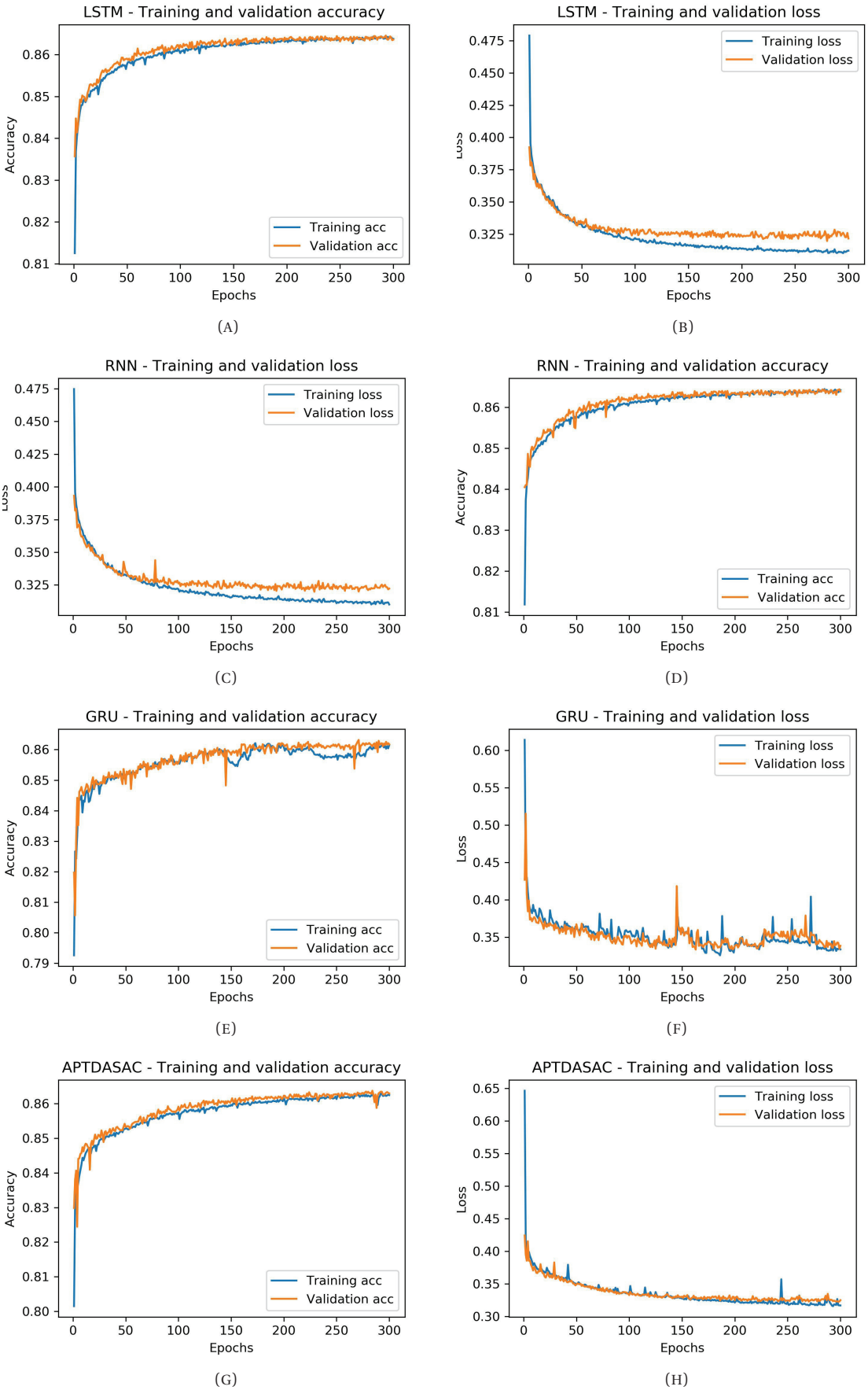
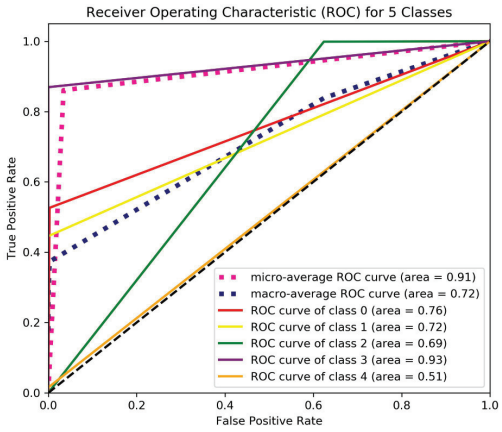
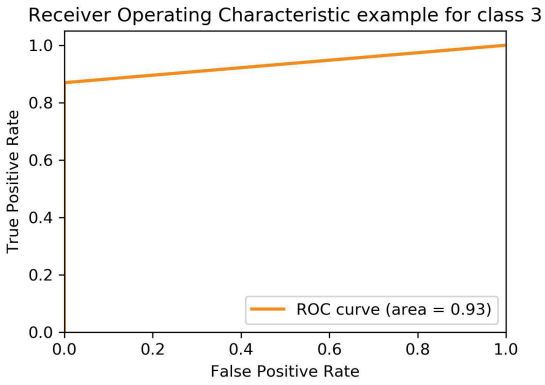


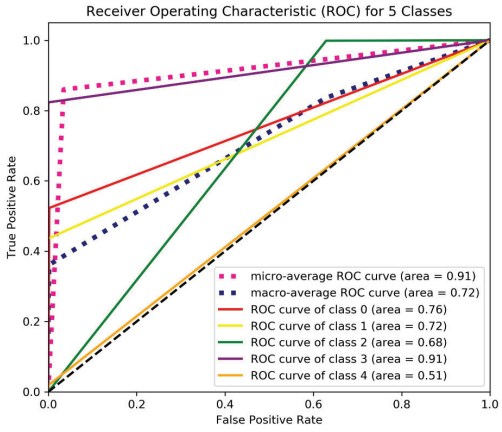
FIGURE 6.10: Validation Accuracy and Loss rate against Epochs for NGP Dataset



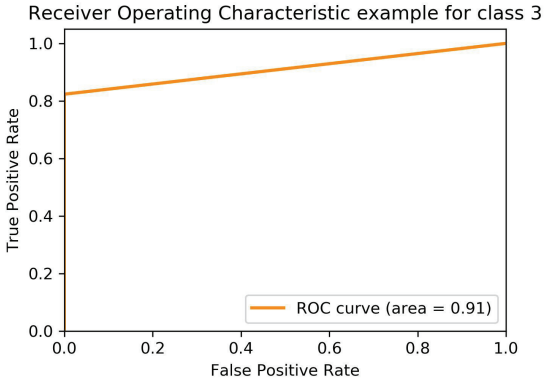
(A) AUC-ROC curve of all classes - LSTM



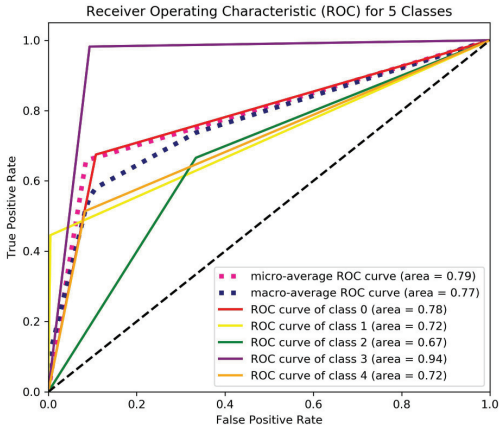
(B) AUC-ROC curve for class 3 - LSTM



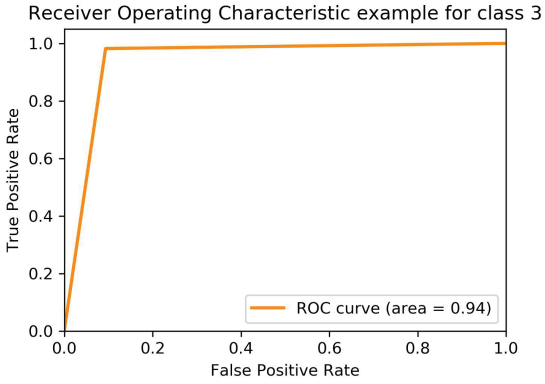
(C) AUC-ROC curve of all classes - APTDASAC



(D) AUC-ROC curve for class 3 - APTDASAC

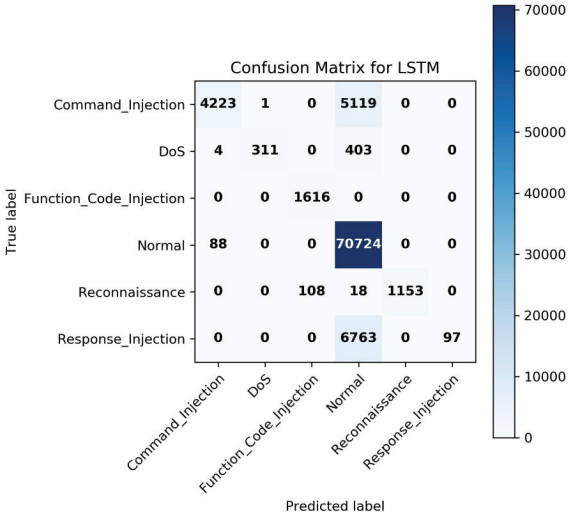


(E) AUC-ROC curve of all classes - ML-DT

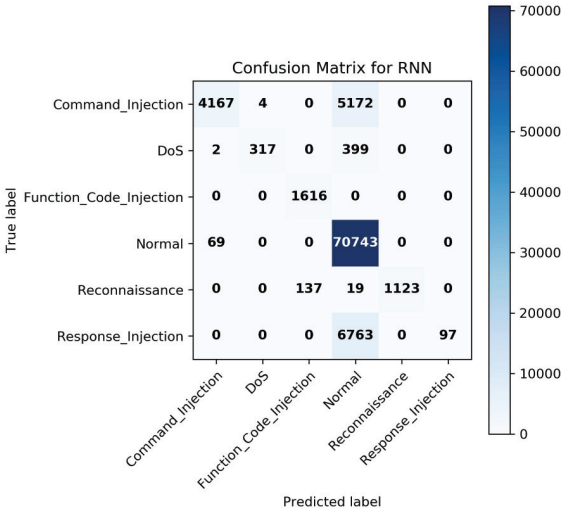


(F) AUC-ROC curve for class 3 - ML-DT

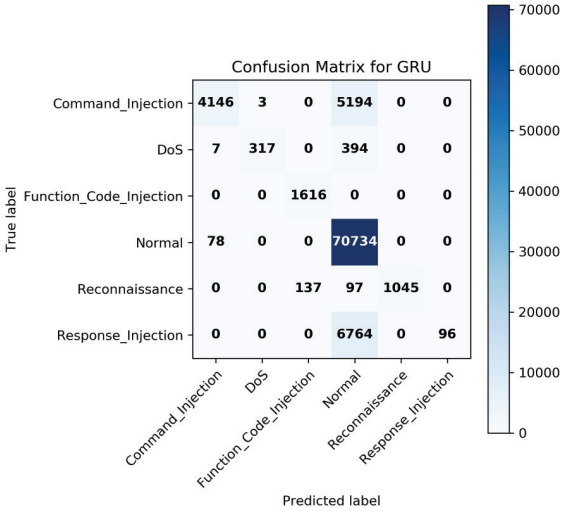
FIGURE 6.11: The visual Representation of AUC-ROC graph for all the main four attack types and normal records for of all the classes and minority ***"class 3"*** for NGP dataset



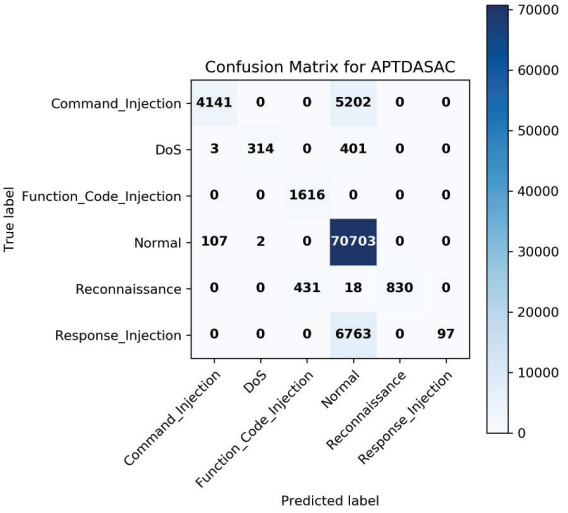
(A)



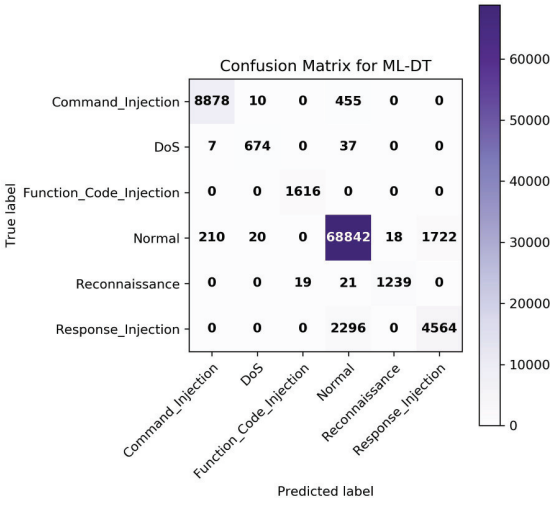
(B)



(C)



(D)



(E)

FIGURE 6.12: Confusion Matrix of NGP_6 Dataset Records

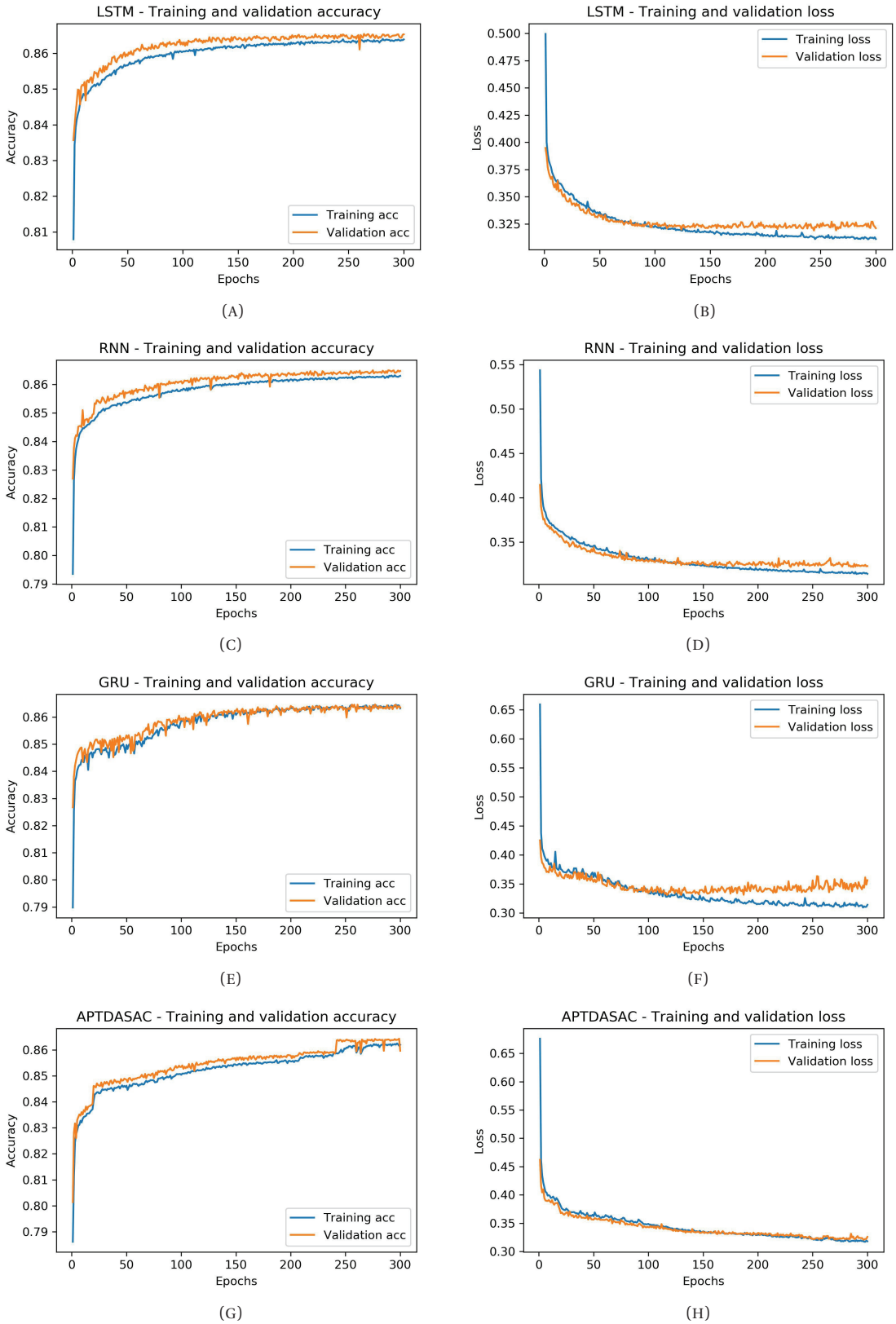
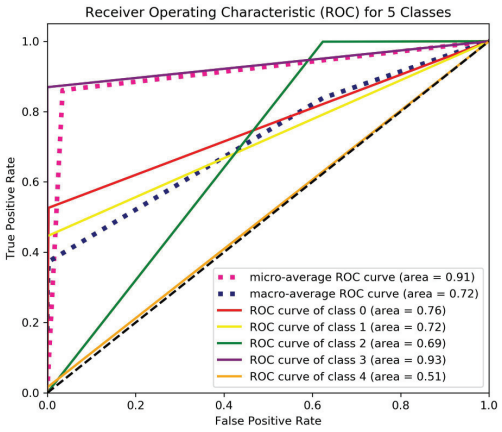
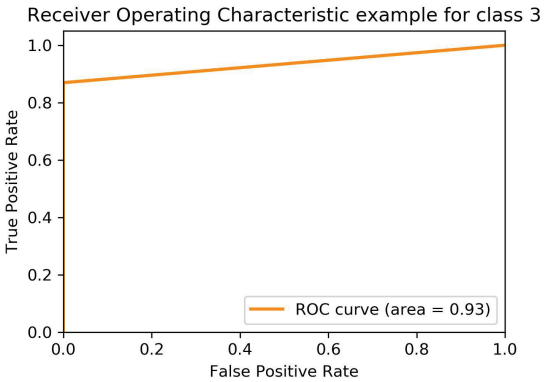


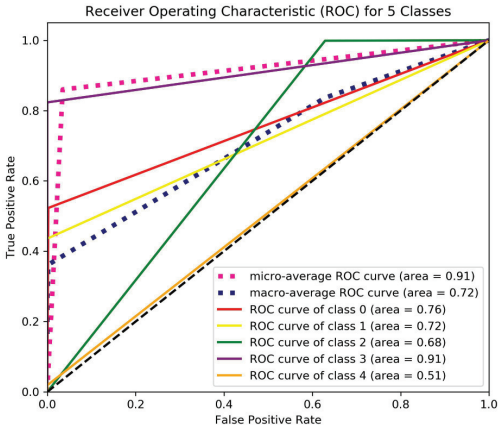
FIGURE 6.13: Validation Accuracy and Loss rate against Epochs for NGP_6 Dataset



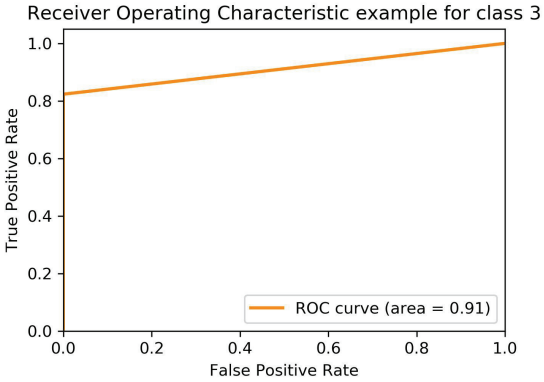
(A) AUC-ROC curve of all classes - LSTM



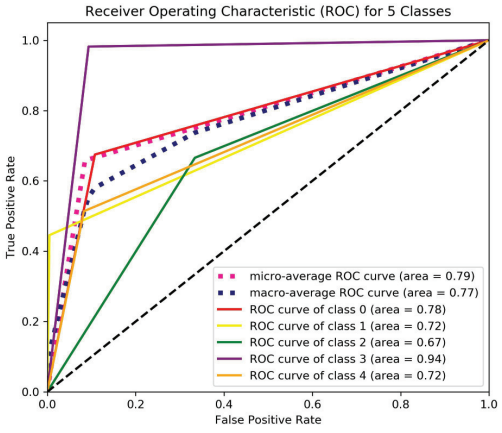
(B) AUC-ROC curve for class 3 - LSTM



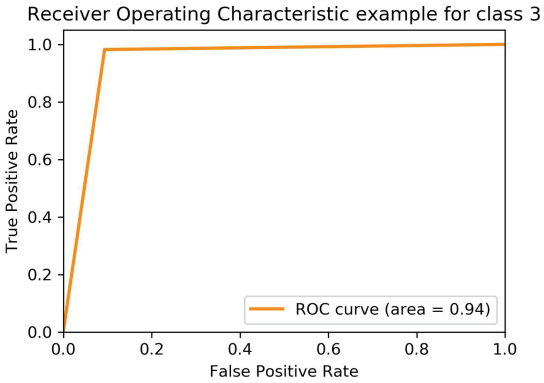
(C) AUC-ROC curve of all classes - APTDASAC



(D) AUC-ROC curve for class 3 - APTDASAC



(E) AUC-ROC curve of all classes - ML-DT



(F) AUC-ROC curve for class 3 - ML-DT

FIGURE 6.14: The visual representation of AUC-ROC graph for all the attack types and normal records and minority ***"class 3"*** for NGP_6 dataset

From the first case study of Sub-section [6.3.1], two experiments were conducted. The classification report shown in Figures [6.2]-[6.3], contains the individual *precision, recall* and *f1-score* of each attack categories, where the *average weighted precision, recall* and *f1-score* are 88%, 86%, & 82% respectively for APT_{DASAC} and 77%, 66% & 70% for ML-DT algorithm from the first experiment. While obtained 88%, 86% & 81% on NGP_6 dataset with overall probability average prediction accuracy of **86.30%** & **86.36%** with loss as **0.32%** shown on Table [6.5] for both experiments. ML-DT achieved *weighted average precision, recall* and *f1-score* of 95% on the second experiment. Since the main interest is to capture all the individual attack steps, it can be seen that out of the total predicted actual values of each of the individual attack, the model was able to correctly predict an overall average weighted recall of 86% for which 88% were actually predicted as attacks. However, a closer observation of the individual precision and recall of each of the algorithms indicates that the model did struggle to correctly identify response_injection, even though it achieved 100% precision when considering four ICS attack types as contained within the dataset.

Further investigation into case study one of Sub-section [6.3.1], with the validation accuracy plotted against loss rate represented in Figures [6.10] and [6.13] and the plotted AUC-ROC, shown in Figures [6.11] and [6.14] for the first and second experiments shows a clear overview of the overall performance as can be visualized in these Figures. Although, there are some noticeable spikes in some of these validation accuracy or loss against epochs graph as shown in Figures [6.10] and [6.13]. However, considering the individual ROC curve achieved, which is above 50 percentile, and with micro & macro-average ROC curve of 91% & 72% respectively obtained, indicates that the model performs well for each individual attack step since the average *ROC curve* is above 50%.

In the second case study [6.3.2], the generated Confusion Matrix [6.15] of this case study shows the predicted values and the actual true values of all the four attacks group and normal instances of the data for each of the implemented algorithms. The visual observation of the individual Figure [3.11], shows a clear picture of the number of instances of the R2L, U2R and Probes with lower connection records while DOS and normal network event appear to have more frequency connection records. Those group with more records are learnt properly without confusing their identity while those with fewer connection records during training did not show good true positive rate and precision as it was had to identify them. This shows the difficulty in capturing rare attack type with low network records. The dataset contains many examples for "neptune" that belongs to DOS attack type, "satan" attacks belongs to Probe, and "normal" but fewer examples of the others.

The implemented model was able to achieve a significant *precision, recall, f1-score* and *overall average accuracy* of 100% with *TPR* of 99.98% and *FPR* of 0.05%. Also, achieved a macro/micro average roc of 96%/1.00 and an *overall average prediction accuracy* of 99.92%, which indicates a good model performance result. However, as pointed out earlier in 6.3.2, this data contains unintended patterns which may have contributed to algorithms to learn differences among patterns so easily, making the results and the detection rate very high as presented in Tables [6.3], [6.4] and [6.5].

Figure [6.16] is the generated validation accuracy plotted against loss rate, while Figure [6.17] is the generated *AUC-ROC* curve from the experiment showing the consolidated individual single graph representing the respective *AUC* curve graph. However, considering the individual *ROC* curve achieved, with 82% lowest *ROC* curve for class 4, and micro/macro-average *ROC* curve of 100% & 96% respectively. These results indicates that the model performed very well for each individual attack step. Substantially, this may be largely attributed by the unintended patterns, which may have contributed to algorithms learning the differences among patterns so easily which resulted to high *AUC-ROC* curve.

The authors in [1], acknowledged the need to investigate application of data re-sampling techniques to manage imbalance data distribution (third case study), to ascertain the impact of applying SMOTE oversampling techniques with focus on detecting the minority rare attack class "Worms" among the multi-class attacks samples. Applying SMOTE techniques did slightly improve the overall average detection rate of Worms attack from 0.03%, 0.06% & 0.09% to 32.50%, 35.50% & 23.2% for the individual algorithms implemented.

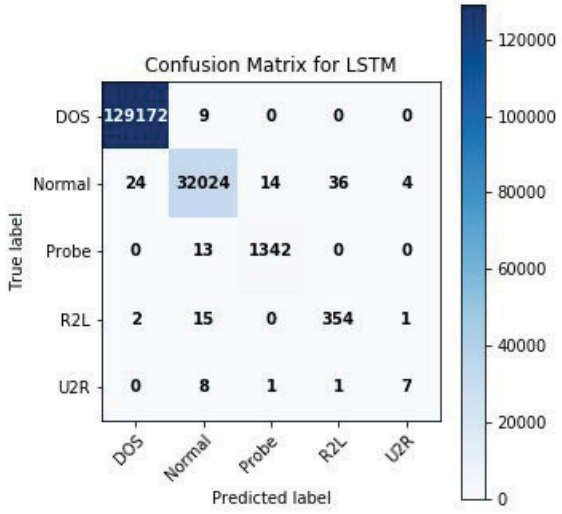
While the *ROC* curve of minority class of interest "Worms" also improves from 50% to 59%, with micro & macro-average *ROC* curve of 73% and 65% respectively, and maximum average mean accuracy of 81.02%. However, the achieved results are still below average, this demonstrate that uneven data distribution as discussed by arthur's in [279], [242] and [280] is not the only factor that could impact model performance since high classification error discussed in [242], also contributed to the poor model performance as presented in [1].

Nevertheless, in this third case study, stacked ensemble-RNN variants for APT_{DASAC} approach were implemented on UNSW-NB15 dataset without any re-sampling techniques applied to establish the capability of this model to detect each individual attack within an enterprise environment. Figure [6.5], gives report account of implementing APT_{DASAC} on the case study three [6.3.3], showing the individual precision, recall and f1-score of each attack categories, where the average weighted precision, recall and f1-score are 83%, 82%, & 80% respectively for APT_{DASAC} and 77%, 76% & 75% for ML-DT algorithm, with overall probability average prediction accuracy of **82.19%** and loss of **42%** shown on Table [6.5].

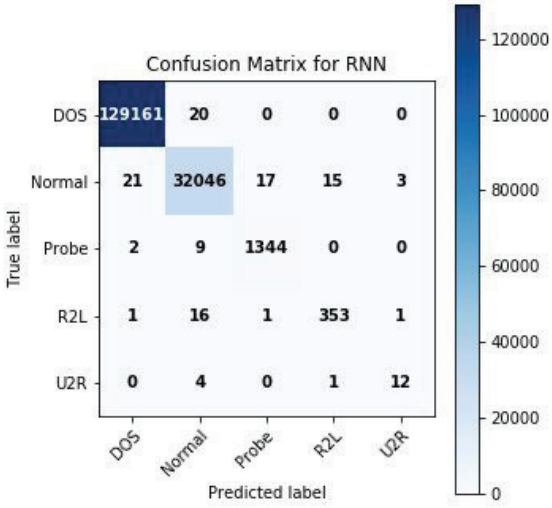
Also implemented the same approach with ML-based approach by replacing, stacked ensemble-RNN variants with ML-DT algorithm on UNSW-NB15 data, and achieved 75.02% weighted average accuracy rate. The overall average detection accuracy rate of 82.19% were achieved as recorded in Table [6.5], for APT_{DASAC} performance which is slightly lower than 86.36% & 99.92% achieved with NGP and KDDCup'99 dataset respectively.

ROC curve scores were generated for the individual class labels, which is shown in Figure [6.20]. The average curves of the classes were evaluated and consolidated into a single graph representing their respective *AUC* curve and obtain *micro-average ROC curve area* of **89%** and *macro-average ROC curve area* of **80%** for the APT_{DASAC}, and obtain *micro-average ROC curve area*/ *macro-average ROC curve area* of 86%/78% respectively from implementing ML-based approach. It is evident from Figure [6.20] that the identification of APT attack detection in step 4 stage has the ROC curve area of **99%** from both models, this is largely attributed to the number of connection record exhibited in this stage, while the class 1 stage has the lowest *ROC* curve area of 52% for APT_{DASAC} model, and 53% for ML-based model.

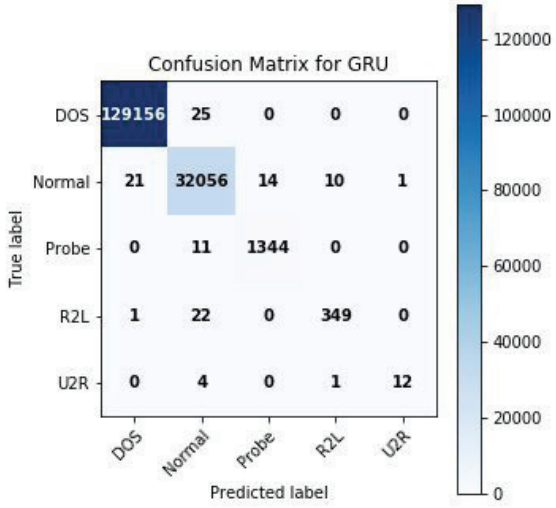
Furthermore, in the fourth case study, the designed APT_{DASAC} approach was implemented on APT_alerts_dataset.db to establish the capability of this model to detect each individual detectable attack step within this simulated APT alerts data with eleven attack steps of which eight alerts are detectable. These detectable alerts align to four detectable steps of APT lifecycle (see Figures [3.4] & [6.7] for the eleven alert steps and the four detectable attack steps group respectively). Figure [6.6], gives report account of implementing APT_{DASAC} on the fourth case study discussed in Sub-section [6.3.4]. The individual precision, recall and f1-score of each attack categories, where the average weighted precision, recall and f1-score are 87%, 87%, & 87% respectively for APT_{DASAC} and 56%, 68% & 61% for ML-DT algorithm, with overall probability average prediction accuracy of **86.73%** and loss of **0.69%** shown on Table [6.5].



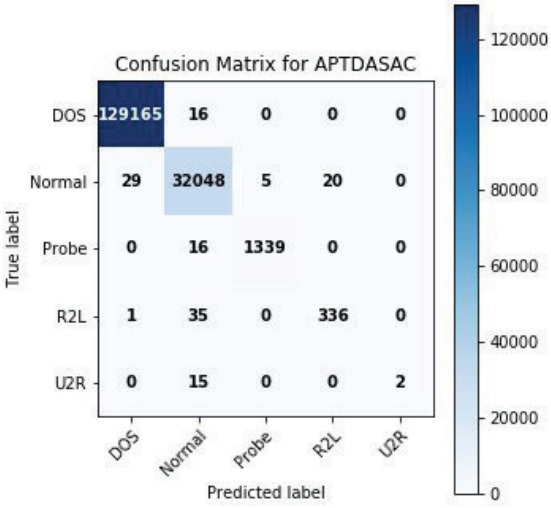
(A)



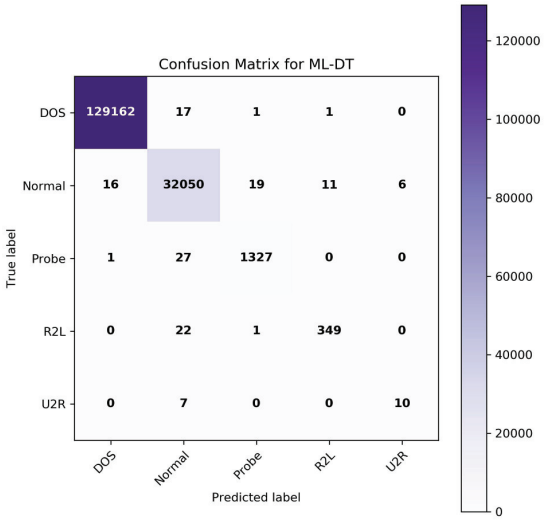
(B)



(C)



(D)



(E)

FIGURE 6.15: Confusion Matrix of KDDCup'99 Dataset Records

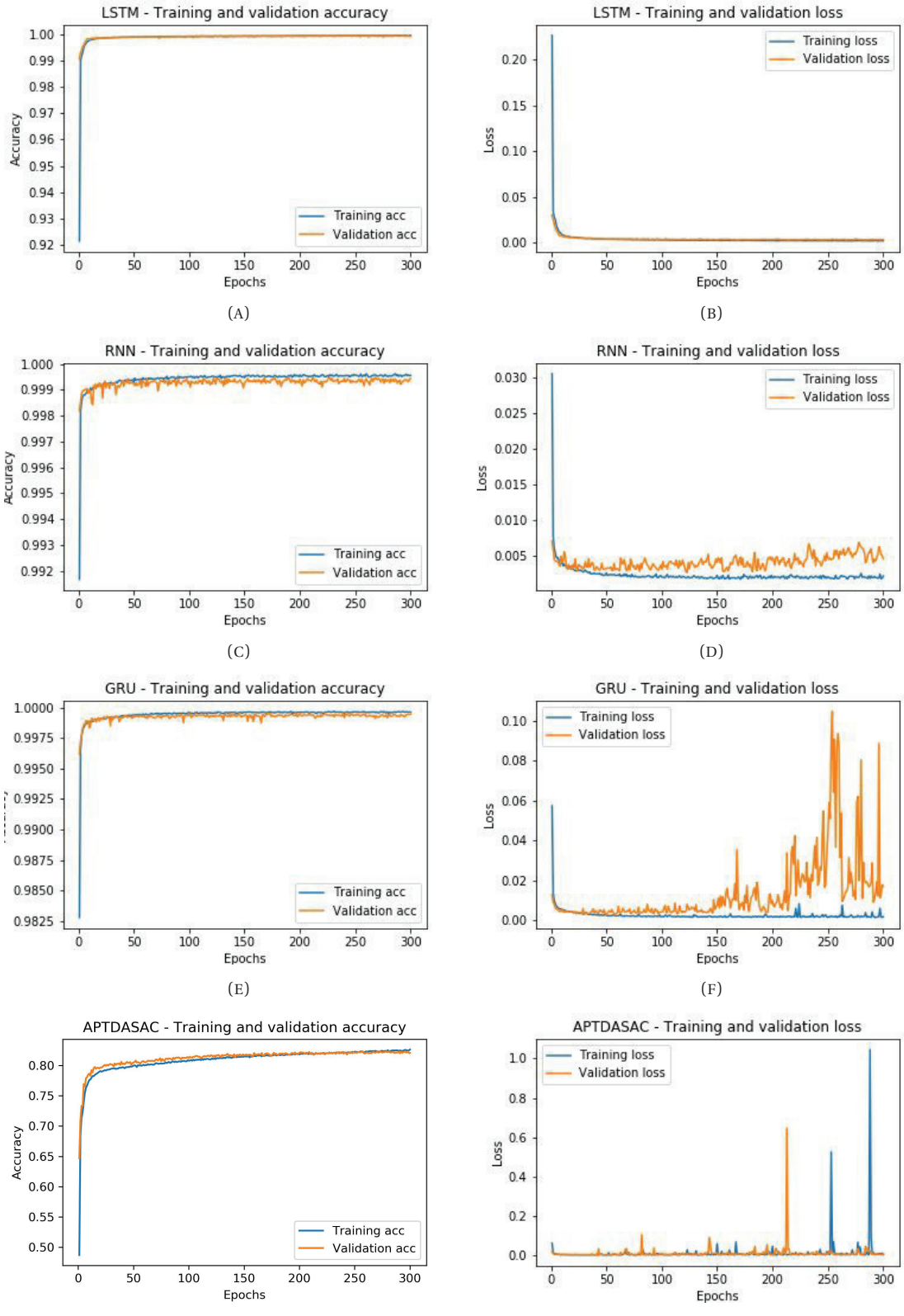
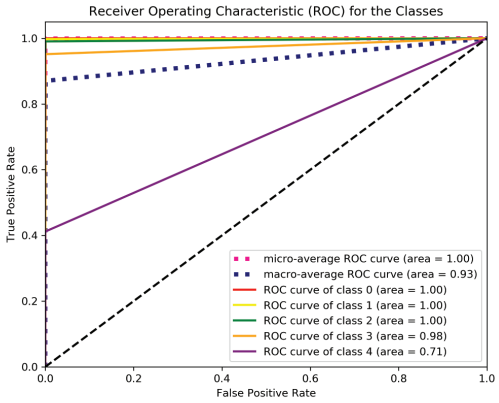
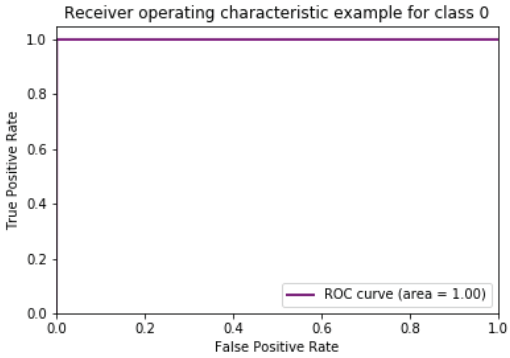


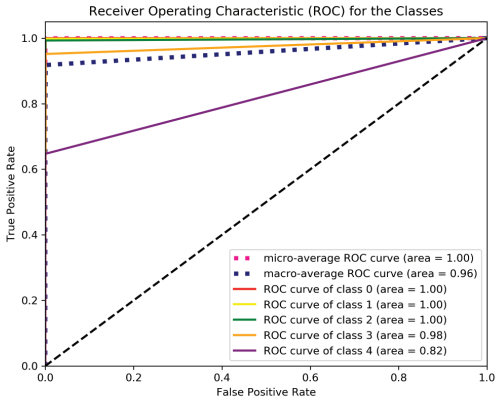
FIGURE 6.16: Validation Accuracy and Loss rate against Epochs for KDD-Cup'99 Data



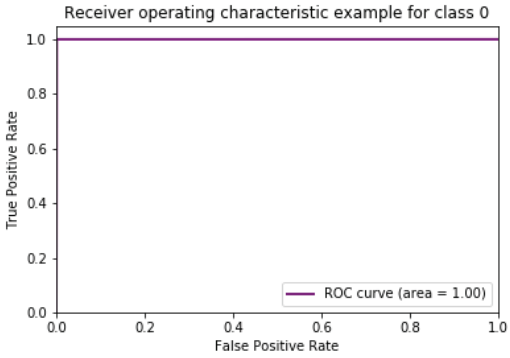
(A) AUC-ROC curve of all classes - LSTM



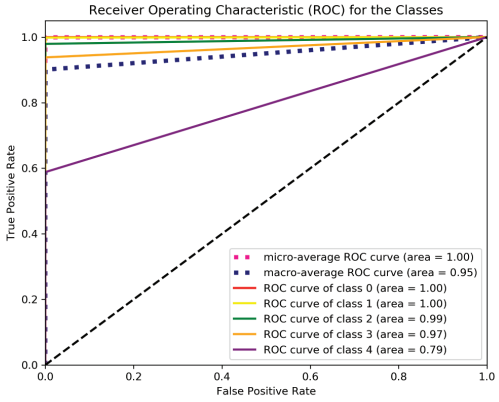
(B) AUC-ROC curve for class 0 - LSTM



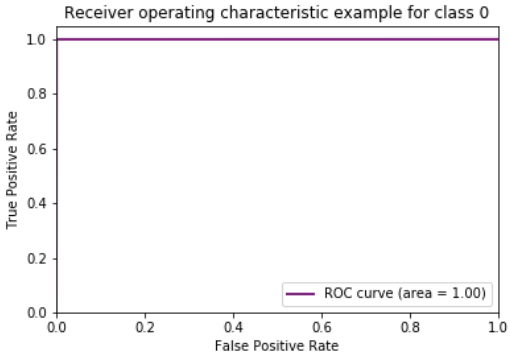
(C) AUC-ROC curve of all classes - APTDASAC



(D) AUC-ROC curve for class 0 - APTDASAC



(E) AUC-ROC curve of all classes - ML-DT

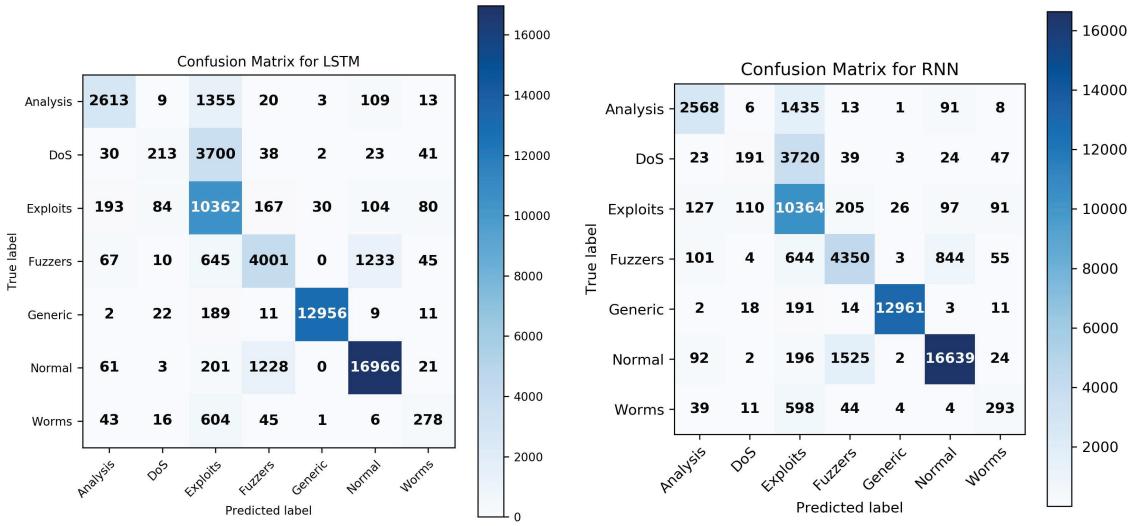


(F) AUC-ROC curve for class 0 - ML-DT

FIGURE 6.17: The visual Representation of AUC-ROC graph for all classes and ***"class 0"*** for KDDCup'99 Data

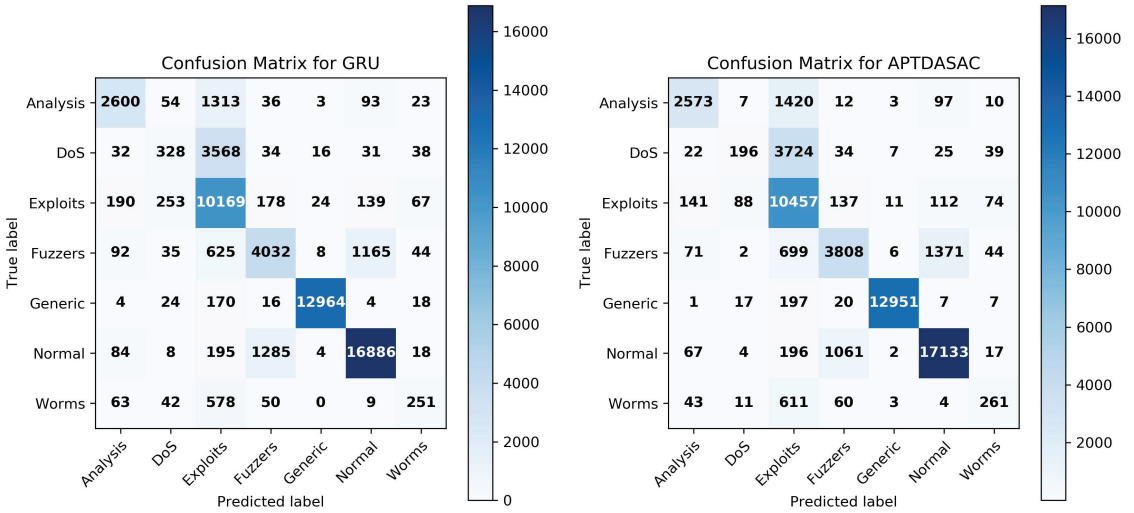
ROC curve scores were generated for the individual detectable step group, which is shown in Figure [6.22]. The average curves of the detectable steps were evaluated and consolidated into a single graph representing their respective *AUC* curve and obtained *micro-average ROC curve area* of **91%** and *macro-average ROC curve area* of **91%** for the APT_{DASAC}, and obtained *micro-average ROC curve area*/ *macro-average ROC curve area* of 82%/80% respectively from implementing ML-based approach. It is evident from Figure [6.22] that the identification of APT attack detection in class 2 stage has the ROC curve area of **96%** & **95%** from both models, this is largely attributed to the number of connection record of 1000 exhibited in this stage, while the class 3 stage has the lowest *ROC* curve area of 87% for APT_{DASAC} model, while ML-based model obtained it's *ROC* curve area of 70% for class 0 followed by 72% for class 3.

The measured matrix values are also recorded in Tables [6.3]-[6.5]. As can be seen from the reported results, all the implemented algorithms returned high average accuracy and detection rate for the four cases in identifying attacks and classifying attack types. The average overall result from the proposed model result was also compared to ML-based model result in Sub-section [6.5.1]. APT_{DASAC} results clearly seem to achieve good results since the weighted average of the *ROC* curves area tends towards 1, (see: Figures [6.11], [6.14], [6.20] and [6.17]. A high area under the curve represents both high recall and high precision, an ideal model with high precision and high recall will return many positive good results, with all results mostly correctly identified and correlated.



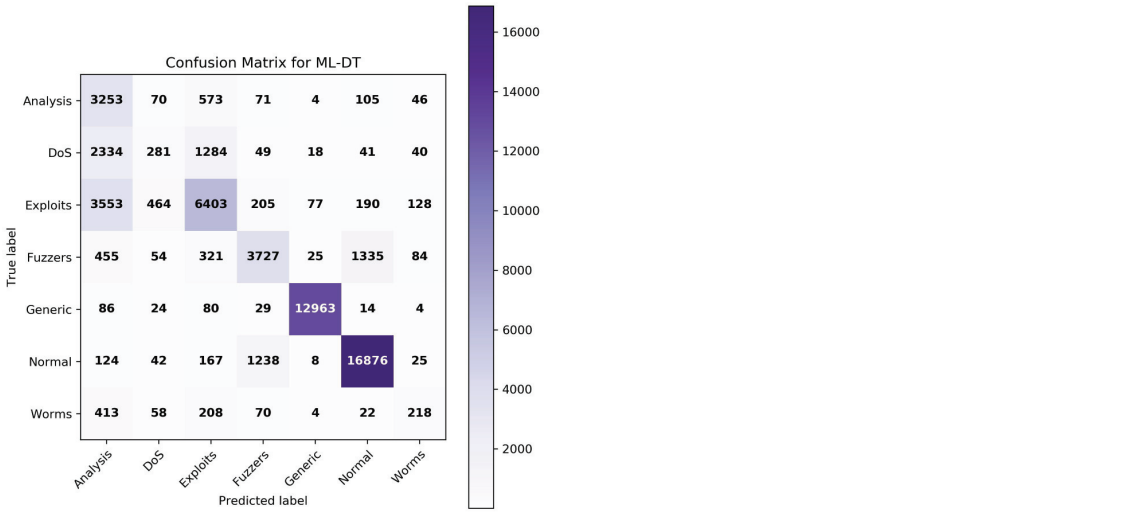
(A)

(B)



(C)

(D)



(E)

FIGURE 6.18: Confusion Matrix of UNSW-NB15 dataset Records

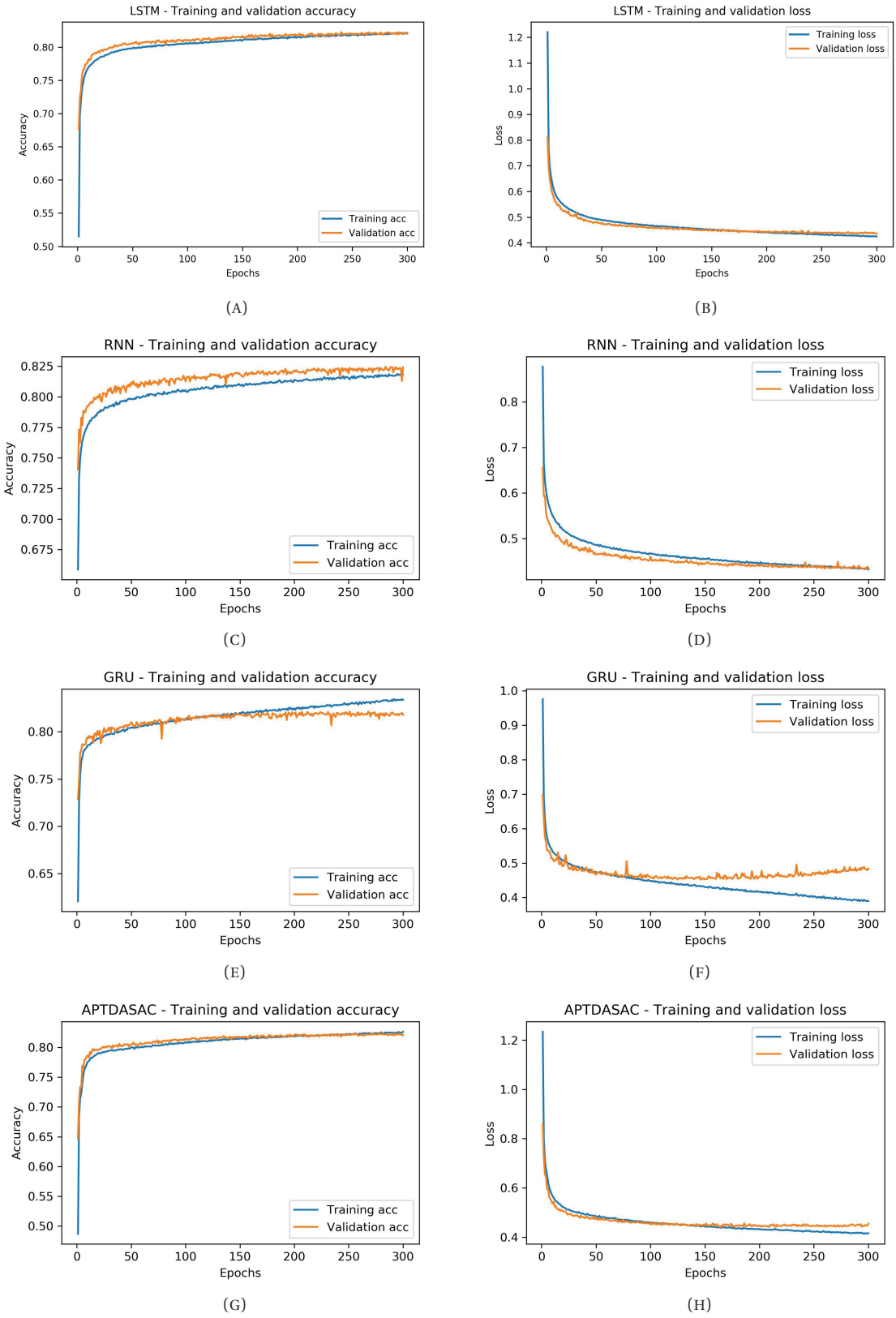
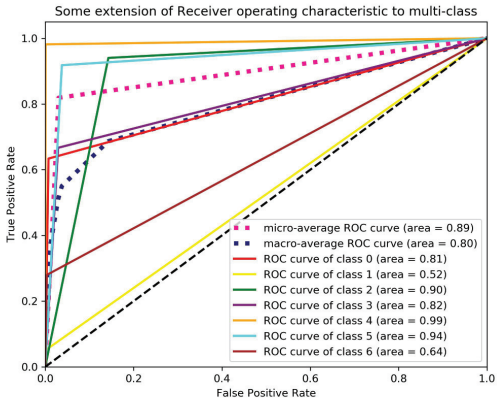
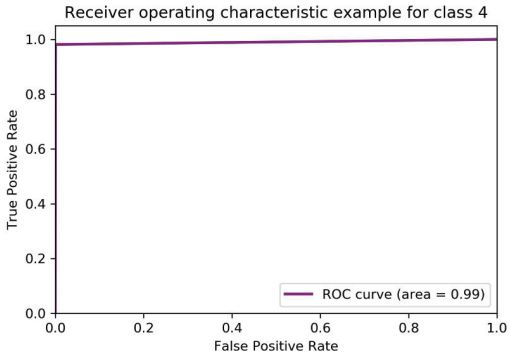


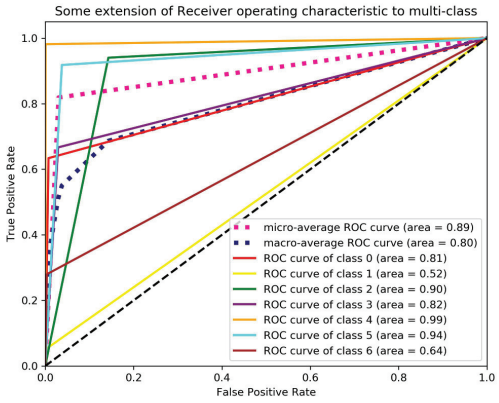
FIGURE 6.19: Validation Accuracy and Loss rate against Epochs for UNSW-NB15 Data



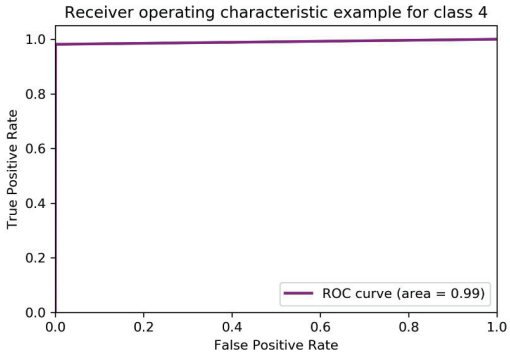
(A) AUC-ROC curve of all classes - LSTM



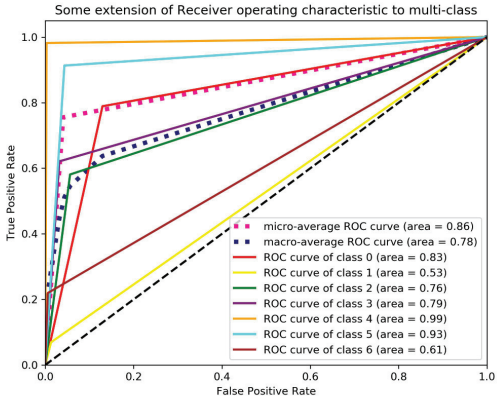
(B) AUC-ROC curve for class 4 - LSTM



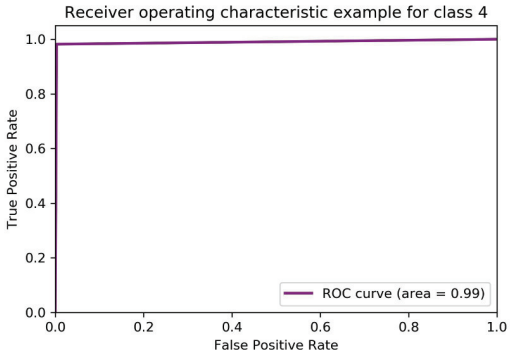
(C) AUC-ROC curve of all classes - APTDASAC



(D) AUC-ROC curve for class 4 - APTDASAC



(E) AUC-ROC curve of all classes - ML-DT



(F) AUC-ROC curve for class 4 - ML-DT

FIGURE 6.20: The visual Representation of AUC-ROC graph for all classes and ***"class 4"*** for UNSW-NB15 Dataset

(A) Confusion Matrix - APT_{DASAC}

(B) Confusion Matrix - ML-DT

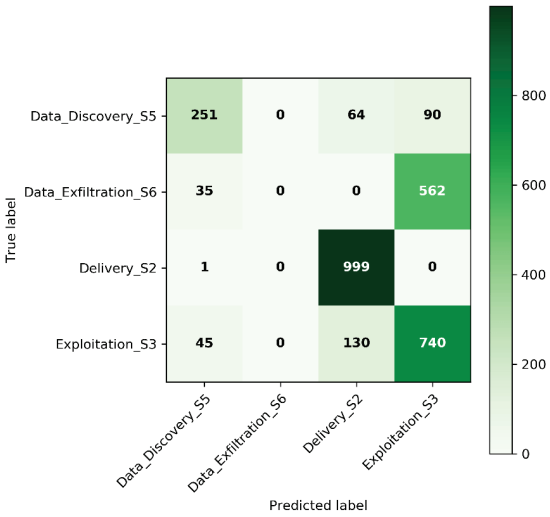
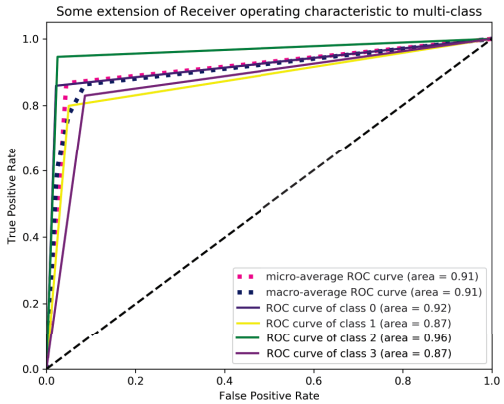


FIGURE 6.21: Confusion Matrix for APT_alert_dataset Records

(A) AUC-ROC curve for -APT_{DASAC}



(B) AUC-ROC curve for class 4 - APT_{DASAC}

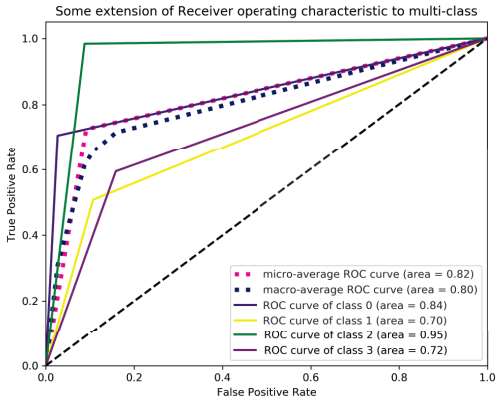


FIGURE 6.22: The visual representation of AUC-ROC graph for all the detectable attack steps for APT_alert_dataset

6.5.1 Model Performance Comparison

The developed framework has been implemented to validate the capability of this thesis proposed model to detect attacks at different stages and correlate these attack steps into detectable step of APT lifecycle. This developed model has demonstrated a good detection capability. From Figures [6.2]-[6.5]. The APT_{DASAC} and ML-DT approach achieved a significant weighted average f1-scores of 82%/70% & 82%/95%, 81%/95%, 80%/75%, 100%/100%, and 87%/61% for the first, second, third, and fourth case studies respectively.

Although, both APT_{DASAC} and ML-DT model achieved exceptionally high f1-score & weighted average detection rate on individual attack detection, it can be seen that both approaches did struggle in detection and classifying some of the individual attacks such as "Response Injection" attack in case study one, "U2R" in case study two and "DoS" in case study three. This is a serious concern as a good model is expected to have the capabilities of detecting all stages accurately and in a timely manner with high recall and precision, since detecting a single attack does not infer detection of a full APT case scenario.

Also, Table [6.4]-[6.5] contains the experimental results summary. From Table [6.3], case study one [6.3.1], APT_{DASAC} achieved a higher overall average accuracy of 87% in the first experiment with ML-DT achieving 66%, whereas in the second experiment, the reverse was the case, where the achieved results are 86% & 95% for APT_{DASAC} and ML-DT respectively with the result of ML-DT slightly higher. While in case study two, APT_{DASAC} and ML-DT achieved 82%/76% and 100% for both approaches in case study three. Also in the last experiment, both model achieved overall average detection accuracy of 87% for APT_{DASAC} and 68% for ML-based approach.

Furthermore, Table [6.4] shows that ML-DT achieved the lowest TPR of 31.84% & 46% when applied to UNSW-NB15 and NGP_5 datasets, while APT_{DASAC} did achieve 88.18% & 96.28% on UNSW-NB15 and NGP_5 dataset respectively. Though both achieved a very high TPR on KDDCup'99 data with insignificant FPR of 0.05%, while obtained 96.90 TPR with 0.04 FPR for APT_{DASAC} and 87.80 TPR for ML-based approach. This demonstrates good model detection performance capabilities.

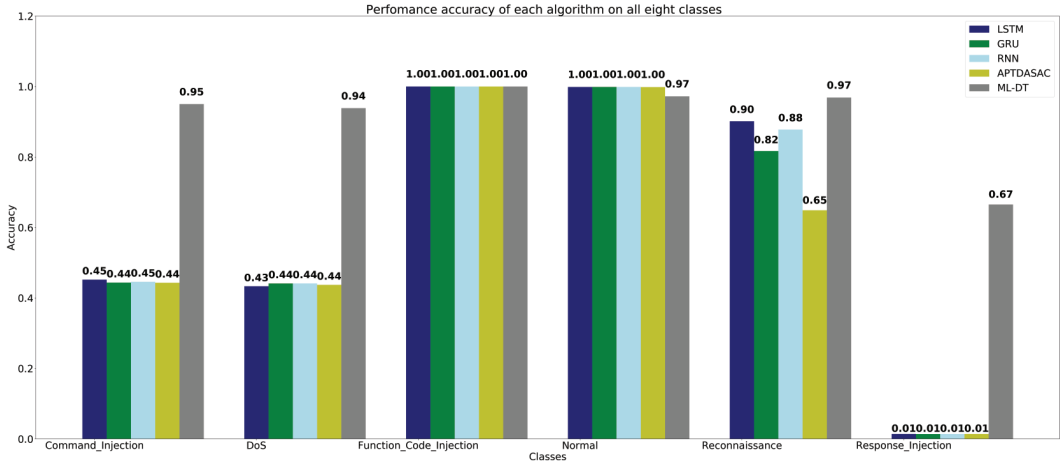


FIGURE 6.23: Algorithm Comparison - NGP_6 Data

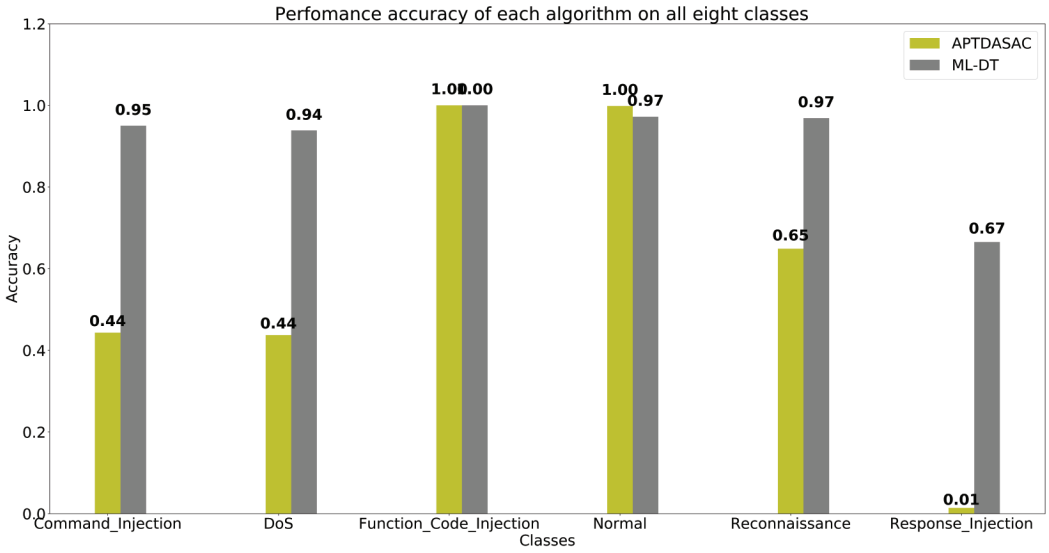


FIGURE 6.24: APTDASAC and ML-DT Comparison - NGP_6 Data

Figures [6.23], [6.25] and [6.27] contains the comparative summary result of the individual algorithms scores, while Figures [6.24], [6.26], [6.28], and [6.29] presents the overall probability detection performance comparison of applying APT_{DASAC} and ML-DT approaches across the four case studies. These figures, shows clear comparison result of each model individual attack detection from each case. In the sense that from Figure [6.23] and [6.24], ML-DT approach outperformed the APT_{DASAC} approach, which only seems to detect more of normal event, although both models achieved 100% in *"function_code_injection"* attack detection. While in Figures [6.27], [6.28], and [6.29], it can be seen that APT_{DASAC} model did outperform ML-DT based approach in all individual attack probability detection performance especially in the case of detecting *"Exploit"* attack.

Furthermore, both model did show good detection capabilities in Figures [6.25] and [6.26], with ML-DT showing a stronger detection capability of 59% on *"U2R"* and APT_{DASAC} yielded slightly higher score of 99% on *"Probe"* attack. Nevertheless, considering all the evaluation metrics used in this thesis, on the application domains and the achieved results, a closer observation and comparison of each model from one domain to other and from one model to another, it can be seen that a model performing very well in one network environment, does not infer that the same model will perform very well, when applied to a different network with completely different network activities.

Pragmatically, both "APT_{DASAC}" and "ML-DT" model, have proven to be good candidates for model development depending on area of application. One can argue from this study that performance of attack detection techniques applied can be influenced by the nature of network transactions with respect to the domain of application. Also, a particular technique applied may be good in detecting a particular step of attack than the rest of the steps as can be seen in Figure [6.29], where ML-based approach noticeably performed poorly in detecting the last stage of APT lifecycle of data exfiltration, while APT_{DASAC} did achieved 80% of detection of tor-connection attack. Hence, functionality, robustness and resilience of any operational devices, such as critical infrastructure state and performance are influenced by the safety and security measures in place which is specific to the system, technology or domain of application, this is directly controlled by the network transaction of that specific domain or systems.

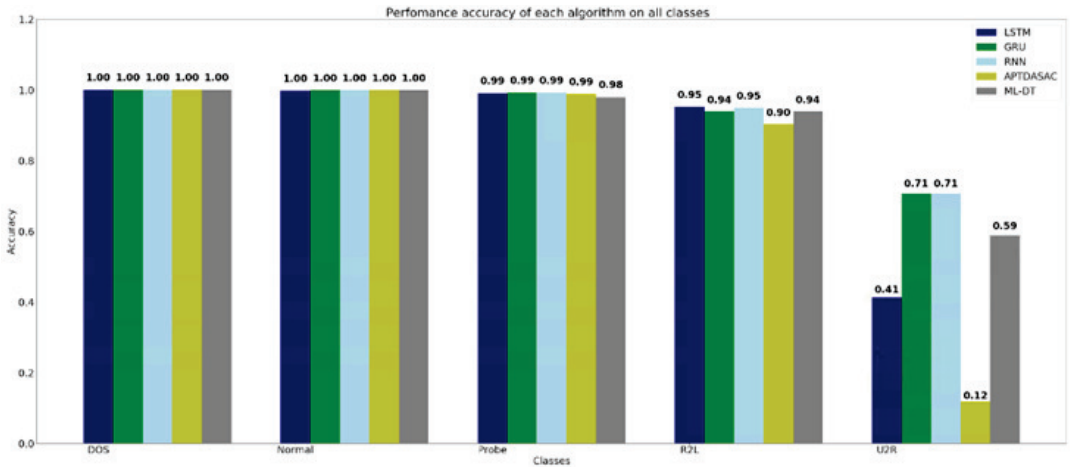


FIGURE 6.25: Algorithm Comparison - KDDCup'99 Data

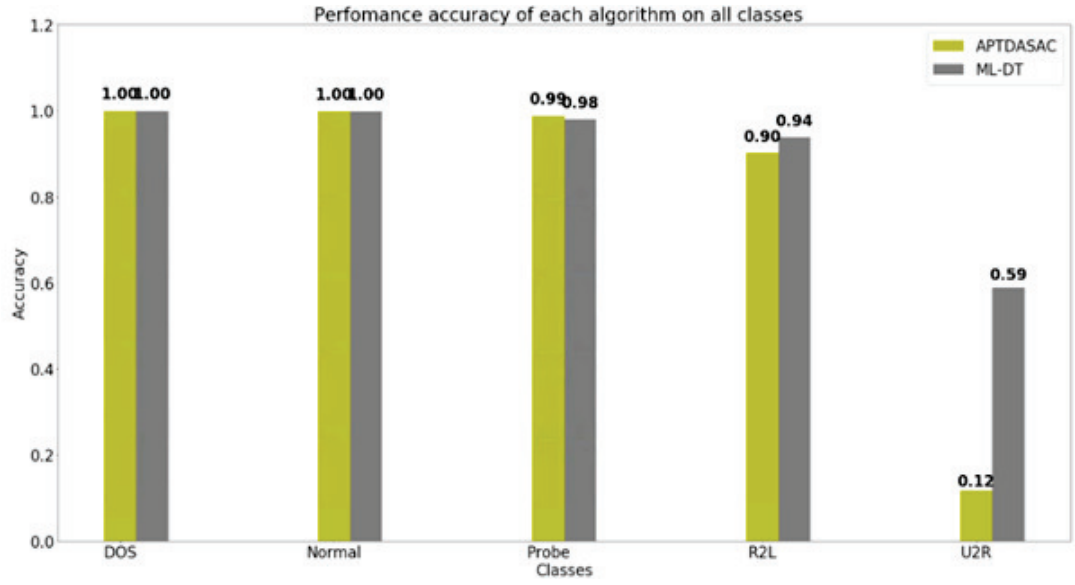


FIGURE 6.26: APT_{DASAC} and ML-DT Comparison - KDDCup'99 Data

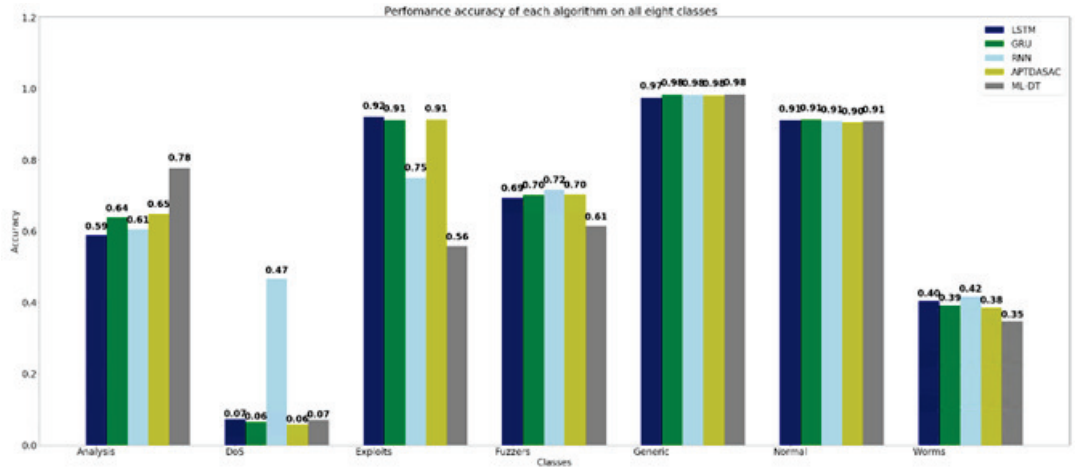


FIGURE 6.27: Algorithm Comparison - UNSW-NB15 Data

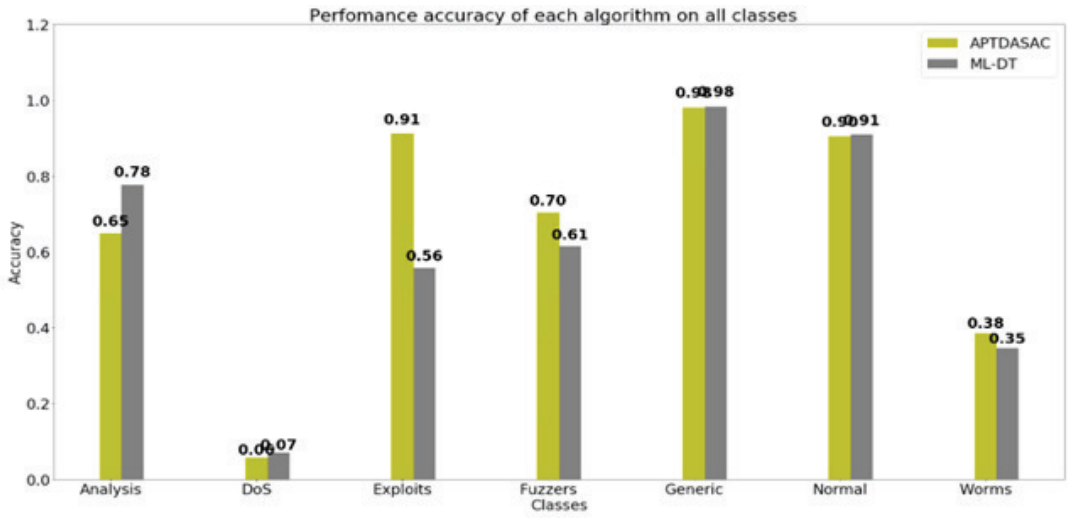


FIGURE 6.28: APT_{DASAC} and ML-DT Comparison - UNSW-NB15 Data

6.5.2 APT_{DASAC} Comparison to other Existing APT Detection Systems

The ability of an IDPSs to detect every single active attack within a system is a security issue. Diverse approaches have been proposed and successfully implemented to address these security issues. Few of which has been reviewed for the purpose of this study as recorded in Tables [3.1] and [6.6].

This study previous work in [51], proposed an approach using deep neural networks for APT multi-step detection which utilized stacked LSTM-RNN networks to learn data features and capture the malicious patterns of APT activities using KDDCup'99 dataset. This approach achieved a detection rate of 99.90% as recorded in Table [6.6]. Also in [5], a framework named APT_{DASAC} based on stacked ensemble-LSTM variants, that takes into consideration the distributed and multi-level nature of ICS architecture and reflect on the four main SCADA cyber attacks which are interception, interruption, modification and fabrication as recorded in [181] was proposed to demonstrate the ability of this approach to detect different stages of APT activities. This approach achieved an overall detection rate of 85% for NGP dataset and 93.67% for UNSW-NB15 dataset. Also, when ML-DT was implemented, obtained 95% on both NGP and UNSW-NB15 datasets. While this thesis, achieved an overall probability accuracy of 86% with f1-score of 82.01%, TPR of 96.28% & FPR of 0.62% for NGP dataset. Also achieved 86.73% overall probability detection of accuracy with f1-score of 87%, TPR of 96.90% & FPR of 0.04% on APT_alerts_dataset.db.

MalNet detection approach proposed in [39] to capture the malicious file structure and code sequence patterns using DNNs for malware detection. This study utilized stacking ensemble to join two networks' discriminant result with an extra metadata feature, and achieved accuracy rate of 99.88% and TPR of 99.14% with a false FPR of 0.1%. Considering a case of non-linearities in communication data flow in an AGC system, authors in [238], implemented a stacked RNN-LSTM model as a detector and classifier to detect FDI attacks in AGC systems and achieved an accuracy rate of 94%. [101], implemented 6-layer DL classification approach for APT attack classification and detection over 10 epochs, to classify attacks from normal class and achieve 98.85% accuracy with FPR of 1.13%. However, this study is based on two classes as all the attacks are grouped under attack against normal events.

Work in [237] proposed an APT attack detection method based on BiLSTM and GCN to analysed network traffic into IP-based network flows. This approach achieved 98.24% of normal IPs and 68.89% of APT attack IPs using Malware Capture CTU-13 data warehouse dataset. However, this study focused on detecting two class; normal IP and APT infected IP. Again, the authors in [302], tackled APT attack detection using network flow-based C&C detection method to detect the hidden C&C channel of unknown APT attacks and achieved an *f1-score* of 96.80%. However, detection rate for this approach was not provided. Also, an APT detection framework based on an enhanced SNN algorithm using semi-supervised learning approach on LANL dataset to scores suspicious APTs-related activities at three different stages of APT attack life cycle was proposed by author [74]. A high weight rank is given to hosts that depict characteristics of data exfiltration with the belief that data exfiltration is the main APT attack. However, this study faced a higher computational overhead cost.

Nevertheless, all reviewed approaches on this study demonstrated high significant APT attack detection capabilities, however, none of these approach used the same dataset or the same metrics measure as can be seen on Tables [3.1] and [6.6]. Most of these studies such as [101], only focused on two classes (attack and normal event), depend on an agent to detect elementary attack and has high FPR – TeminAPTor proposed in [42]. May require significant expert knowledge to set up and maintain the it's functionality as the case of Statistical APT Detector proposed in [246] or may focus on one or two features added to their own generated data for PADASYN evaluation [239]. These makes it difficult to compare or rank the performance of these approaches. Additionally, the unavailability of a standard acceptable framework, dataset or suitable public accessible dataset is a huge challenge in cyber security research community, making it unfavourable to compare an APT detection systems performance that may help to chose an appropriate model for any given domain.

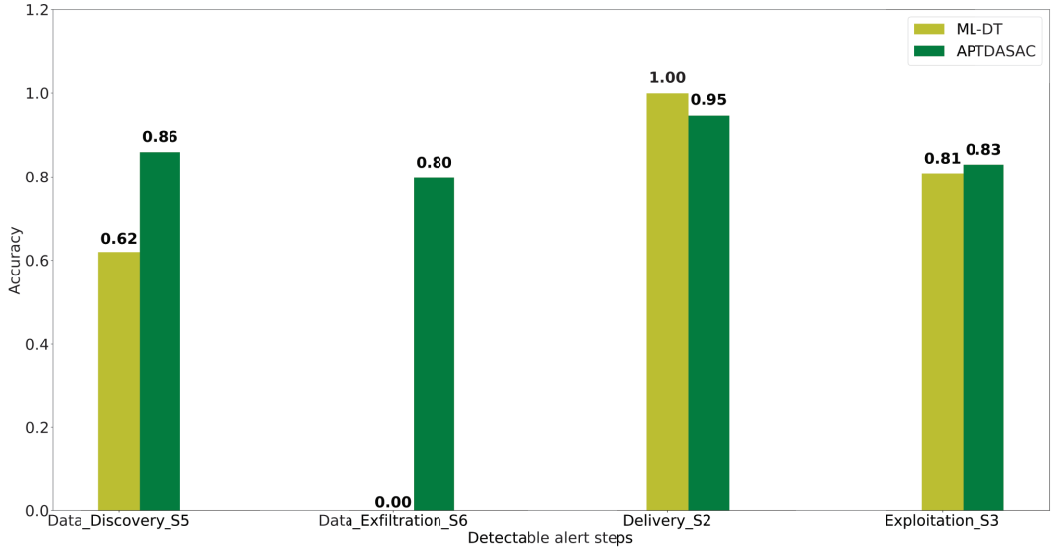


FIGURE 6.29: APT_{DASAC} and ML-DT detection capability on the four detectable steps of APT_alert_dataset.db

TABLE 6.6: APT_{DASAC} Comparison to other Existing APT Detection Systems

Proposed APT Detection Systems / Reference	Used Approaches	Data Source	TPR (%)	FPR (%)	f1-score (%)	Overall Prediction Accuracy(%)
APT_{DASAC} (Current study)	DL Techniques	NPG	96.28	0.62	82.01	86
		UNSW-NB15	88.18	2.15	79.57	82
		KDDCup'99	99.98	0.05	99.93	100
		APT_alerts_dataset.db	96.90	0.04	87.00	86.73
ML-DT (Current study)	ML Techniques	NPG	97.61	1.42	94.6	95
		UNSW-NB15	31.84	7.05	75.02	76
		KDDCup'99	99.99	0.05	99.92	100
		APT_alerts_dataset.db	87.80	3.38	62.00	68.22
Hidden Markov [231]	ML Techniques	APT_alerts_dataset.db	?	?	?	66.50
Ensemble $LSTM_{variants}$ [5]	DL Techniques	NGP	96.5	0	82	85
Enhanced $SNN_{Algorithm}$ [74]	Semi-supervised learning approach	LANL	?	?	?	90.5
BiLSTM & GCN [237]	Network flow analysis	Malware Capture CTU-13 data warehouse	?	?	?	68.89 (APT IPs attack)
Network flow based on C&C detection method [302]	DL Techniques	Contagio blog malware	?	?	96.8	?
Stacked $_{LSTM-RNNmodel}$ [51]	DL Techniques	KDDCup'99	100	0	99.9	99.9
A 6-layer DL model [101]	DL Techniques	NSL-KDD	?	1.13	?	98.85
MalNet [39]	DL Stacking ensemble	Generated grayscale image data	99.14	0.1	?	99.88
Stacked $_{RNN-LSTMmodel}$ [238]	DL Techniques	?	?	??	?	94

6.6 Mitigation of APTs at different Stages in the Lifecycle

CCERT-EU [303] reported an increase of APT attacks of 60% against EU institute in 2022 compared to 2019. Statista in [304], also reported that the APT protection market is expected to exceed 15 billion U.S. dollars by 2026 as a result of APT attack increase in number. Also, according to Gatewatcher, 93% of businesses across the UK, France and Germany engage in the detection and discovery of APTs regardless of the business size [305]. Predominantly, most organisations use in-house operation to address this APT attacks, while 19% outsource their APT response to a service provider or managed security service provider (MSSP), this did increase to 23% for France that uses a partner for APT protection.

6.6.1 APTs Lifecycle Mitigation Approach

As highlighted in [11], detection of a single APT step, does not infer detection of APT scenario. APT actors use several TTPs to achieve their goal. These TTPs /or steps can get obfuscated by the actor to evade the detection mechanism. The authors in [306], believed that finding attackers source and the associated campaign behind the threats can lead to an ideal approach for realizing best defense mechanism against APTs in timely manner. The author proposed a multi-view fuzzy consensus clustering model for attributing malicious payloads to their associated APT actor. The author conducted 4000 experiments, extracted 12 combinations of views for the attribution task using five APT families' payloads. Comparison analysis of a single-view against multi-view approach and achieve 95.2% accuracy for their proposed approach in attributing the malicious payloads to their actors.

Lifecycle of an APTs:

Reconnaissance: Reconnaissance activities which is the first step of APT attack step is seen as the prerequisite to most of the cyber attacks, especially APT attacks. This step is used to exploit networks IP addresses and the static nature of MAC addresses in a wireless network [307]. IDS/IPS devices could be used to protect network perimeter by not allowing access to the internal entities. Though application of some of these measures are limited to internal traffic and insider attacks. To address this network perimeter reconnaissance activities, authors in [307], proposed incorporation of MAC addresses randomization of wireless networks into iOS 8, IP address randomization based on DHCP [308], & IP address randomization based on NAT in [309] are proposed. However, authors in [78], took full potentials of address randomization to propose address randomization technique called random host address mutation (RHM), for protecting enterprise networks from both external and internal reconnaissance and scanning. The RHM approach thwart and slows down attacks progress in the network by distorting adversarial reconnaissance, thereby increasing attack detectability.

Delivery by download: is a vital part of APT lifecycle through which malware can be delivered automatically through exploitation of software vulnerability, weak points of protocols or may be by tricking user into communicating with the malware. Example: an email with malicious attachment file that can be used to deliver malicious malware. This type of attack step can be mitigated through users' awareness, require software patches to reduce vulnerability and the chances of system user been used as an instrument to deliver malware [310], and malware inspection and content filtering [55].

Scanning Attack (Exploitation): Work in [223] presented an adaptive scanning technique, named DNS-cache based scanning, that exploits local DNS cache to bypass prior detection methods. The scan detection system (SDS) monitors the in and out flows of

an enterprise network subnet and detects scanning probes based on the correlation of flows with preceding DNS query, thus decreasing the TTL values of DNS resource records (RR). The incorporation of TTL reduction mechanism enhances the effectiveness of this approach to 96% accuracy and also denies the attacker an opportunity to piggyback on cached DNS records which enables them to bypass been detected.

TABLE 6.7: *Measures for Mitigating APT Lifecycle*

APT Lifecycle (Steps)[148] & [2.6]	Attack Methods [148], [155]	Alert_Types [232]	Defense Measures
Step 1 Reconnaissance / Weaponization	Social Engineering, OSINT [11] Active Scanning [155] Search Open Technical Databases [155] Gather Victim Org Information [155] Search Closed Sources [155]		- User awareness [105], [311], [312] [78] - MAC address randomization [307] - IP address randomization based on DHCP [308], - IP address randomization based on NAT [309] - Incorporation of MAC addresses randomization of wireless networks into iOS [307] - Pre-compromise [155]
Step 2 Delivery	Spear Phishing, Watering-hole Command and Scripting Interpreter [155] Scheduled Task/Job [155]	disguised_exe_alert hash_alert domain_alert	- Requires software patch to reduce vulnerability [310] - Malware inspection, Content filtering, Blacklisting [55]
Step 3 Initial Intrusion / Exploitation	Domain fluxing Content Injection [155] Replication Through Removable Media Valid Accounts [155].	ssl_alert ip_alert domain_flux_alert	- Analyses of failed DNS query against the threshold for - DNS query failures from the same IP address [292], [293] - Restrict Web-Based Content [155] - Encrypt Sensitive Information [155].
Step 4 Lateral movement / Operation	Privileges Escalation [155], Malware, Vulnerabilities exploitation		- HetGLM (Thwart attack during LM phase) [83], - LMTracker (Access authentication denial) [313], - Access Control Listing, Firewall, Password Control [314]
Step 5 Data Discovery / Collection	Command & control channels Device Driver Discovery [155]	scan_alert	- Command & control messages of an attack can be detected by analyzing data that leaves the network - Encryption use control [55] - Access Control Listing, Firewall, Password Control [314]
Step 6 Exfiltration / Cover up	- TOR communication network [296] - Utilizing onion routing to direct client's traffic over a circuit of different relays to its destination [297] - Command and control - Email Collection & Input Capture [155]	tor_alert	- ZXAD (Suppressing traffic flow at the Tor exits) [315] - Alerts triggered, Memory forensics [316] - Firewall, Proxy, Encryption use control, blacklisting [55], [314] - Encrypt Sensitive Information [155].

Lateral Movement (LM) attack: Users of compromised system under LM attack are believed to have access and interact with devices they usually would not normally have access to. To track and trace the relational link among network entities, Xiaoqing et al, in [83] proposed HetGLM, an LM detection technique using heterogeneous graph neural network through discovering malicious links. HetGLM is used to apprehend authentication activities that deviate from normal network patterns by profiling each network entity. This is achieved by constructing a heterogeneous authentication graph based on various types of logs, then specifies meta-paths to represent complex relations among network entities. This is realised through application of meta-paths based sampling strategy and attention mechanism to capture rich semantics among devices and users. Again, another technique called LMTracker is designed to detect LM attack path based on construction of heterogeneous graph that generate representation vectors for LM path using event traffic and logs. The application of this approach, detect LM attack and preserve the attack path relationship as stated by the author, [313]. The preserved path logs is utilized by the security professionals to analyse attack activities.

Data Discovery & Collection: Attacker's goal is to conceal their presence and maintain access within the target's network in other to find sensitive data without been discovered. If access is lost, APTs finds its way to vulnerable system, explores target's network and searches for sensitive data of interest. Any gathered data is encrypted to hide its true identity and evade detection while been transferred from target system through the redundant copies of C&C channel created [110], debugger evasion, and group policy discovery [155]. This type of APT attack technique is based on abuse of system features, mitigation of this step of APT is hard and cannot be easily mitigated with preventive controls [155]. However, monitoring the LDAP queries for abnormality with filters for *groupPolicyContainer* and high volumes of LDAP traffic to domain controllers can be used to detect such attack technique in active directory service.

Data Exfiltration: The TOR communication network is utilized by attacker to transfer stolen information from a staging server to attackers' external servers, either in the form of encrypted packages, password protected zip files or even through clear web mails [11]. Due to the anonymity of Tor communication network, attacker often misuse this to convey attack traffic by exploiting the relay through which traffic exit Tor. Suppressing the traffic flow at the Tor exits may help with early detection of Tor attack thereby improving the overall integrity of information been sent and received. Thus pre-empt blanket blocking of Tor exits. ZXAD, a zero knowledge based private Tor exits misuse detection system that permits identification of part of a high-volume attack of unlinkable connections is introduced in [315]. It operates with low bandwidth and processing overheads. The author believes that this approach only discloses information about user transferring a high volume of traffic through Tor.

The adversaries may also use *interactive command shells technique* to gather sensitive information from *removable media* for exfiltration. This type of attack technique can be mitigated through *data loss prevention* by restricting access to sensitive data, detection of sensitive data that is not encrypted, and monitoring access to removable media such as optical disk drive connected to the compromised system for unauthorized file access attack [155].

6.7 Summary

This chapter, presented the evaluation results achieved from implementing APT_{DASAC} model, a multi-stage framework on four different case studies; The New Gas Pipeline Control System identification of data injection attacks and classification; KDDCup'99 data - a wide variety of intrusions activities simulated within military network environment, UNSW-NB15 data containing a hybrid of synthesized attack activities of the network traffic has been presented and discussed. Also implemented APT_{DASAC} on APT_alerts_dataset to establish the capability of this model to detect each individual detectable attack step within this simulated APT alerts data. Few existing approaches for APT detection as reviewed for this thesis were also compared to APT_{DASAC} framework outcome. Based on the outcome of these case studies, the developed model has also been tuned for better performance outcome. The finding as discussed in [6.5.1], suggest that a hybrid of "APT_{DASAC}" and "ML-DT" model will make a good model for APT attack steps detection, to take advantage of the combination of the strength of both model. Either a model based on DL and ML to achieved a better detection capability since both approach demonstrated significant attacks detection capability. Few suggested mitigative measures to defend against APT lifecycle (steps) are also discussed in this Chapter as summarized in Table [6.7].

Chapter 7

Conclusion

***

7.1 Summary 149
7.2 Limitations and Future Work 152
7.3 Conclusion 154
A.1 Book Chapter 175
A.2 Conferences 175
A.3 Journals 175
A.4 Symposium 175
A.5 Poster 176
C.1 Code Snippets 179
Index 187

The purpose of this thesis was to propose and develop a novel system to detect APT attack steps, and check its effectiveness. In this chapter, the summary of this thesis contributions and conclusion of the findings were presented. Some of the limitations encountered during the course of this study were highlighted and future study areas suggested.

7.1 Summary

APT attack is an intentional targeted malicious attack with a clear targets and goal(s). This type of attack utilizes multiple sophisticated tactics to deliver and exploits their targets. Although various existing IDPSs and firewalls are able to detect and mitigate different types of cyber attacks in a very effective way, on the other hand, cyber criminals are relentlessly developing more and new advanced tactics and techniques to penetrate, evade and exploit their target's network. This type of attack has become a huge challenge to governments, organizations and businesses security systems. Diverse methods of using an approach based on ML or DL algorithms to analyse network traffic activities for anomaly patterns within system operation that could aid an early attack detection and possibly prevent or mitigate APT attacks are well sought after in security research communities.

However, these methods faces several pitfalls such as lack of data with real attack campaign, real time detection capability, able to detect all steps of APT attacks, as most approach focus on detection of a single step which does not infer detection of complete APT scenario. Also been able to correlate network events and align any detected attack

to APT attack steps (lifecycle) over a long period of time, in addition to having satisfactory balance between TPR & FNR for uneven distributed network events (rare attack) with harmonic recall & precision. To manage some of these security system challenges, most recent studies build their own experimental datasets and extract APT attack steps from the built datasets. Using the generated data for model evaluation may yield high detection accuracy, but does not, reflect high efficiency in real time practice.

This study developed a novel multi-stage APT attack detection framework, deployed the developed system to investigate APT attack detection in different domains. I designed and developed an APT attack detection system termed APT_{DASAC} with focus on attack TTPs detection and correlation of these attack steps into detectable attack step within APT lifecycle stages. An approach based on Stacked ensemble deep learning model to detect and correlate these APT attack steps has been applied. The ability of this implemented approach to detect APT attack should be high with low FNR as to determine the performance and effectiveness of this model. Thus, the research questions and objectives outlined in Sub-sections [1.4.1] and [1.4.2] are utilized as a guide to address the research aim stated in Section [1.4]. The APT_{DASAC} model is designed to run in three main stages (see detailed description in Section [4.2]) to detect and predict APT attack steps as follows;

- ☐ *Stage 1: Data Input and Probing Layer*
- ☐ *Stage 2: Analysis & Correlation Layer*
- ☐ *Stage 3: Decision Layer*

7.1.1 Contributions Summary

In this subsection, the contributions and findings from this thesis are summarised:

- **Studied APT attack mode of operation and investigated existing APT countermeasures, defence and detection systems**

  This thesis has provided a thorough review of APT traits and lifecycle in Chapter [2], where an extensive review was carried out to understand APT mode of operation and to identify current state-of-the-art techniques geared towards detecting this type of attacks (see Chapter [3]). This thesis further discussed APT traits, lifecycle and give examples of the most significant confirmed cases of APT attack on CPS devices. Also, the current position of publicly available APT attack datasets for research purposes as presented in Chapter [3.5], where lack of publicly available and accessible data containing real APT campaign as a serious issue was identified.

  Few existing important useful and available security tools such as *Purdue Model*, discussed in Sub-section [2.3.4.1] - proposed as fundamental guide for designing ICS architecture, and *MITRE ATT&CK Framework* in Sub-section [2.3.4.2] are highlighted. The *MITRE ATT&CK Framework* is a publicly available knowledge base repository of adversary TTPs that contains a wealth of knowledge for developing specific attack techniques, detection, prediction and mitigation models. Also, explored DL and ML techniques as useful tools in artificial intelligent with interest on RNN variants. RNN and it's variants has emerged as a powerful approach for DL architecture, generally applicable for time-series data modelling as utilized in this thesis for the purpose of validating study designed framework. The objective one *(RO1)* [1.4.2], in relation to research question one (RQ1) [1.4.1], has been addressed by this contribution in part and it has also been used as part of this study publication in [51].

- **Designed and developed a multi-staged model with ensemble deep learning techniques for APT attack detection named *"APT_{DASAC}"***

  Chapter [4], presented the novel approach for detecting APT attack based on supervised learning approaches. This study proposed and designed a novel framework based on *"APT steps analysis and correlation of APTs lifecycle"* abbreviated as ***"APT_{DASAC}".*** This framework utilized stacked ensemble deep neural networks using RNN variants for realising the multi-staged security detection system. I designed a multi-stage framework with generic settings which can be deployed in different domains for the purpose of safeguarding the system against any form of malicious intrusion activities, this is a great contribution to cyber security community (see: Figure [4.1] of Chapter [4]).

  The rationale behind this design are based on few factors; APT attack is considered as a type of malicious attack that is carried out in multiple stages or steps using different tactics and techniques at different stages to deliver and exploit the network and resources of their target. Considering that, the detection of any single step of any of the attack tactics does not infer detection of an APT full scenario, a detection model should be designed and developed with these multi-steps tactics in mind, such that, the model should have the capability to detect different techniques used to deliver APT campaigns at different stages. The model should also be able to correlate and link each individual technique in other to ascertain APT campaign, thereby reducing the FPR of the detection system. This contribution has addressed the second objective [1.4.2] of this thesis in relation to first and second research questions (RQ2) & (RQ3) [1.4.1)]. The implementation of this designed framework and its design has been published in Springer as a Book Chapter [5].

- **Evaluation and analysis of the performance of the developed framework in different application problem domains**

  In Chapter [6], I evaluated the performance of APT_{DASAC} framework based on stacked ensemble-RNN variants multi-attacks steps detection model. Series of experiments were carried out on four different domains using a historical events data of monitored system network activities to evaluate the performance of the developed system in terms of detecting each individual attack step and correlating these attack steps into attack groups. I also evaluated the overall detection accuracy to demonstrate the probability detection capability of this model. I went further to implement the same framework based on ML approach using decision tree to evaluate the the detection capability of this same framework.

  The findings also highlighted some of the difficulties encountered as a result of uneven distributed nature of huge events elements among classes to hide within a weak signals among huge traffic data resulting to imbalance data. This makes it difficult or harder to detect rare type of APT attack step, presented in [1]. Example: C&C APT attacks signal type are harder to detect as mentioned in [221]. The preliminary result for this thesis suggest that high classification error and class separability problem of minority class has a noticeable impact on model overall performance with respect to application domain. As can be seen from the work in [1], the achieved overall average detection rate for minority class of interest *U2R* in KDD-Cup99 dataset improved from 63.20%, 68.40% & 57.60% to 78.90%, 78.90% & 73.70%, while in the case of detecting *Worms* in UNSW-NB15 dataset, implementing SMOTE approach did make an impact as the overall average detection rate improved from

0.03%, 0.06% & 0.09% to 32.50%, 35.50% & 23.2%. However, the detection rate obtained are still below average, suggesting that a high classification error also contributed to the poor system performance.

Most importantly, the experiments also demonstrated that the strength of a model based on DL and/or ML algorithms within different domains may differ based on the network parameter settings. It can be seen that a model that perform very well in a specific domains does not always imply that it will yield the same good result in another network environment when deployed in a different network with completely different network activities. This suggest that taking an advantage of the combination of the strength of both algorithms detection capability to develop a model based on hybrid of both DL & ML could make a good attack detection model. The research questions *(RQ2), (RQ3)* & *(RQ4)* [1.4.1)] in regards to the third study objective (RO3) [1.4.2] are addressed within this contribution in part.

- **Application of APT_{DASAC} on four Different Domains**

  Application of the developed model to four different domains as analysed and presented in Section [6.3], to demonstrate the designed framework performance and also evaluate the detection capability of APT_{DASAC} approach for defending and safeguarding the system operation by detecting attack steps is a great contribution of this work. However, to the best of my knowledge from the reviewed literature work on existing detection systems, DL detection algorithm based on stacked ensemble RNN variants algorithms has not been applied to handle APT attack multi-steps detection, hence the introduction of APT_{DASAC} model based on DNNs approach for multi-steps attacks detection.

  Series of experiments were carried out, to analyse the performance of this framework in these domains including application to the NGP dataset collected by simulating real attacks and operators activity on a gas pipeline [53], to test the performance with respect to detection capabilities in these domains. This study did consider data injection attacks of gas pipeline, enterprise network, military domain environment network activities and simulated APT alerts data. The achieved results suggested that the implemented approach has demonstrated good attack detection capability. Effective performance of attack detection techniques applied can be influenced by the nature of network transactions with respect to the domain of application. Also, a detection system in place may be good in detecting a particular step of APT attack lifecycle than the rest of the steps. This was demonstrated by the fourth case scenario, where ML-based approach was able to detect 100% delivery step on detecting disguised executable attack . This is the second step of APT lifecycle but first detectable step of APT lifecycle. However, it did struggle with last stage of data exfiltration step in detecting malicious tor connections as can be seen in Figure [6.29].

Most of the reported contributions and results in this thesis have been produced and presented as a poster, symposium, also published in journals and conference proceedings, as well as a book chapter, all listed in Appendix [A].

7.2 Limitations and Future Work

This section highlights few of the limitations encountered over the course of this study, and also, outlines some areas for future work considerations.

7.2.1 Limitations

Some of the limiting factors encountered in this thesis include as explained in Section [3.3] are;

- Lack of standard available public datasets containing real APT attacks campaign, sharing of information, multi-class problem, data imbalance, high classification error and difficulty in comparing detection system performance.
- Running of the proposed model also, faced with GPU and CPU limitation of the computer used in this study as the system intermittently crashes as a result of the large data in use, in addition to the number of training iteration involved. Thus, a high GPU-enabled TensorFlow platform and/or cloud-based system can effectively accommodate a good number of batch size, and a higher number of epochs to successfully train a model and ultimately achieve an effective detection capability.
- Also, the APT_{DASAC} framework has been validated on synthetic data from four different domains. However, it would be highly beneficial to test this framework in real time and on real APTs steps data. Nevertheless, such data is not easily accessible, as such, lack of relevant publicly available data sources contributes to the main reason for using synthetic data for this study.

7.2.2 Future Work

The work in this thesis has also demonstrated that implementation of the developed framework can be very useful for threat detection. With that in mind, there are a number of improvements that can be made to improve the system detection capabilities of this APT_{DASAC} framework. There are a number of difficulties and challenges that made APT attack detection approaches highly inefficient. However, the current APT detection systems face serious drawback in several ways, these includes but not limited to

- Achieving real time APT campaign detection, detecting all steps (tactics) of APT attack, able to manage large data in real time as a result of unevenly distribution of APT steps events elements among network events, having a suitable balance between false positive rate and false negative rates, and been able to correlate the network activities spanning over a long period of time as to determine full APTs scenario.
- Future work can be improved by utilizing an algorithm capable of managing big data, handling uneven distributed network events, incorporated with the use of search grid technique to loop through different settings to produce the best hyperparameter settings suitable for that particular system. Improvement can also be made for the system to process the events in real time, and also be able to detect any attack steps in real time over a long period of time.
- Furthermore, since the experiments undertaken have demonstrated that the strength of a model based on DL and/or ML algorithms within different domains may differ based on the network transactions, network parameter settings and capability in detecting a particular step. A hybrid framework based on DL and ML that will take advantage of the combination of the strength of both algorithms' detection capability to develop a hybrid model could make a good attack detection model so as to compliment each other's strength in building a robust system capable of detecting all possible APT steps

- Considering the negative impact and substantial financial loss caused by APT attacks, and the fact that the current intrusion detection and prevention methods still struggle to detect APT in real time, there is a serious need for continuous research in improving the existing method and research new approaches and techniques for detecting, mitigating APT attack, safeguarding systems. Considering that the oil and gas sector fuels every aspect of our daily life, the protection of this critical infrastructure cannot be over emphasized. We cannot afford to underestimate the consequences of such attacks on the operations and systems that power our lifestyle.

7.3 Conclusion

This thesis has addressed the problem of detecting APT attack as a multi-steps attack campaign type. I developed and implemented a multi-staged attack model for handling multiple attack detection scenario. The implemented model utilized ensemble techniques for optimizing detection accuracy through combining network results on RNNs variants due to it's capabilities as highlighted in Section[1.5]. The achieved result demonstrated that using stacked ensemble model, configured with appropriate parameter settings such as activation, optimisation, and loss functions; as well as the correct use of data pre-processing will make a good choice towards developing a system capable of dealing with this type of attack that are been executed in different stages in a real-time dynamic problem environment. Thus, ensemble techniques with the probability combiner are used to combine the approaches used to learn the model, the final decision of the model is made based on the outputs of all the implemented approaches, where the strength of each individual approach compliment the weakness of the other approach.

The first case study, achieved an *average weighted precision, recall* and *f1-score* of 88%, 86%, & 82% respectively for APT_{DASAC} and 77%, 66% & 70% for ML-DT algorithm from the first experiment. While obtained 88%, 86% & 81% on NGP_6 dataset with overall probability average prediction accuracy of **86.30%** & **86.36%**, with loss as **0.32%** shown in Table [6.5], for both experiments. ML-DT achieved *weighted average precision, recall* and *f1-score* of 95% on the second experiment. While in the second case study, the *precision, recall, f1-score* and *overall average accuracy* of 100% with *TPR* of 99.98% and *FPR* of 0.05% were obtained with an *overall average prediction accuracy* of 99.92%. In the third case study, the average weighted precision, recall and f1-score are 83%, 82%, & 80% respectively for APT_{DASAC} and 77%, 76% & 75% for ML-DT algorithm, with overall probability average prediction accuracy of **82.19%** and loss of **42%** as shown on Table [6.5]. Lastly, on the fourth scenario, the average weighted precision, recall and f1-score is 87%, for the three measure when APT_{DASAC} is applied, and 56%, 68% & 61% for ML-DT algorithm. Both achieved overall probability average prediction accuracy of **86.73%** and loss of **0.69%** and **68%** for APT_{DASAC} and ML-DT respectively as shown on Table [6.5].

Also, the summary of individual attack step detection result recorded in Table [6.2], shows that APT_{DASAC} achieved a significant and commendable detection rate when predicting each individual APT detectable steps (A, B, C,& D). It achieved **93.50%, 80.98%, 85.19%, & 80.90%** *detection_rate* for *disguised_exe_alert*, *ip_alert*, *scan_alert* & *tor_alert* demonstrating the capability of this framework to detect each individual detectable step of APT attack step, thus highlighting APT campaign.

Considering different results obtained from the application domains, the implemented approach showed a significant attack detection capability and has demonstrated that performance of attack detection approach applied in any domain, can be influences by the nature of network transactions/events with respect to the domain of application. This

suggest that the ability and resilience of operational system state to withstand attack and maintain system functionalities are regulated by the safety and security measures in place, which is specific to that system/devices or application domain. The study also, suggests that a hybrid of model based on combination of algorithms based on "DL" & "ML" approaches with focus on capturing every single step representing each detectable step of APT lifecycle. Also has the capability to correlate those identified steps into group in other to accurately ascertain presence of APT campaign, could make a good model for APT attack steps detection. This is to take advantage of the combined effort of both models, to achieve a more effective detection capability since both approach demonstrated significant attack detection capability. Hence, this thesis main contributions are in the domain of cyber security.

Bibliography

***

[1] Hope Eke, Andrei Petrovski, and Hatem Ahriz. “Handling minority class problem in threats detection based on heterogeneous ensemble learning approach”. In: *International Journal of Systems and Software Security and Protection (IJSSSP)* 11.2 (2020), pp. 13–37.

[2] Amin Hassanzadeh et al. “A review of cybersecurity incidents in the water sector”. In: *Journal of Environmental Engineering* 146.5 (2020), p. 03120003.

[3] Cristina Alcaraz, Gerardo Fernandez, and Fernando Carvajal. “Security aspects of SCADA and DCS environments”. In: *Critical Infrastructure Protection*. Springer, 2012, pp. 120–149.

[4] A Odewale. “Implementing secure architecture for industrial control systems”. In: *Proceedings of the 27th COREN Engineering Assembly, Abuja, Nigera* (2018), pp. 6–8.

[5] Hope Nkiruka Eke et al. *Framework for Detecting APTs Based on Steps Analysis and Correlation*. Springer, 2022, pp. 119–147.

[6] KK Stouffer and Joe Falco. “Recommended practise: Improving industrial control systems cybersecurity with defense-in-depth strategies”. In: *Department of Homeland Security, Control systems security program, national cyber security division* 3 (2009).

[7] Hyung Seok Kim et al. “Design of networks for distributed digital control systems in nuclear power plants”. In: *Intl. Topical Meeting on Nuclear Plant Instrumentation, Controls, and Human-Machine Interface Technologies (NPIC&HMIT 2000)*. Citeseer. 2000.

[8] Abdulmalik Humayed et al. “Cyber-physical systems security—A survey”. In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 1802–1831.

[9] Grigoris Tzokatziou, Leandros Maglaras, and Helge Janicke. “Insecure by design: Using human interface devices to exploit SCADA systems”. In: *3rd International Symposium for ICS & SCADA Cyber Security Research 2015 (ICS-CSR 2015) 3*. 2015, pp. 103–106.

[10] Robert D Larkin et al. “Evaluation of security solutions in the SCADA environment”. In: *ACM SIGMIS Database: the DATABASE for Advances in Information Systems* 45.1 (2014), pp. 38–53.

[11] Hope Nkiruka Eke and Andrei Petrovski. “Advanced Persistent Threats Detection based on Deep Learning Approach”. In: *2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS)*. IEEE. 2023, pp. 1–10.

[12] Brendon Harris and Ray Hunt. “TCP/IP security threats and attack methods”. In: *Computer communications* 22.10 (1999), pp. 885–897.

[13] Robert M Clark et al. “Protecting drinking water utilities from cyberthreats”. In: *Journal (American Water Works Association)* 109.2 (2017), pp. 50–58.

[14] Eric K Kaufman, Samson Adeoye, and Feras A Batarseh. “Leadership for CyberBioSecurity: The Case of Oldsmar Water”. In: (2023).

[15] MITRE. “ATT&CK® for Industrial Control Systems”. In: (2019). URL: https://attack.mitre.org/matrices/ics/.

[16] Steve Morgan et al. “Hackerpocalypse: A cybercrime revelation”. In: *Cybersecurity Ventures* (2016), pp. 1–24.

[17] Paul Dreyer et al. “Estimating the global cost of cyber risk”. In: *Research Reports RR-2299-WFHF, Rand Corporation* (2018).

[18] Cabinet Office. *National Cyber Security Strategy 2016–2021*. 2016.

[19] Tommy Van Steen et al. “What (if any) behaviour change techniques do government-led cybersecurity awareness campaigns use?” In: *Journal of Cybersecurity* 6.1 (2020), tyaa019.

[20] Luciana Obregon. “Secure architecture for industrial control systems”. In: *SANS Institute InfoSec Reading Room* 2 (2015).

[21] James P Anderson. “Computer security threat monitoring and surveillance”. In: *Technical Report, James P. Anderson Company* (1980).

[22] Christopher D McDermott and Andrei Petrovski. “Investigation of computational intelligence techniques for intrusion detection in wireless sensor networks.” In: *International journal of computer networks and communications* 9.4 (2017).

[23] Mohammad Sazzadul Hoque et al. “An implementation of intrusion detection system using genetic algorithm”. In: *arXiv preprint arXiv:1204.1336* (2012).

[24] Jaeseok Kim et al. “An Innovative Automated Robotic System based on Deep Learning Approach for Recycling Objects.” In: *ICINCO (2)*. 2019, pp. 613–622.

[25] Haotian Shi et al. “Automated heartbeat classification based on deep neural network with multiple input layers”. In: *Knowledge-Based Systems* 188 (2020), p. 105036.

[26] S Rego et al. “Screening for diabetic retinopathy using an automated diagnostic system based on deep learning: diagnostic accuracy assessment”. In: *Ophthalmologica* 244.3 (2021), pp. 250–257.

[27] Eyad Elyan, Laura Jamieson, and Adamu Ali-Gombe. “Deep learning for symbols detection and classification in engineering drawings”. In: *Neural networks* 129 (2020), pp. 91–102.

[28] Shuanlong Niu et al. “A novel deep learning motivated data augmentation system based on defect segmentation requirements”. In: *Journal of Intelligent Manufacturing* (2023), pp. 1–15.

[29] Jong Pil Yun et al. “Automated defect inspection system for metal surfaces based on deep learning and data augmentation”. In: *Journal of Manufacturing Systems* 55 (2020), pp. 317–324.

[30] Shengyun Wei, Shun Zou, Feifan Liao, et al. “A comparison on data augmentation methods based on deep learning for audio classification”. In: *Journal of Physics: Conference Series*. Vol. 1453. 1. IOP Publishing. 2020, p. 012085.

[31] Jawad Rasheed et al. “Effects of glow data augmentation on face recognition system based on deep learning”. In: *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. IEEE. 2020, pp. 1–5.

[32] Ali Maina Bukar, Hassan Ugail, and David Connah. “Automatic age and gender classification using supervised appearance model”. In: *Journal of Electronic Imaging* 25.6 (2016), pp. 061605–061605.

[33] Marc Fossi et al. “Symantec internet security threat report trends for 2010”. In: *Volume XVI* (2011).

[34] Gianmarco Baldini and Irene Amerini. “Online Distributed Denial of Service (DDoS) intrusion detection based on adaptive sliding window and morphological fractal dimension”. In: *Computer Networks* 210 (2022), p. 108923.

[35] Yujie Zhou et al. “Application of time series data anomaly detection based on deep learning in continuous casting process”. In: *ISIJ International* 62.4 (2022), pp. 689–698.

[36] Yu Liu et al. “Anomaly detection based on machine learning in IoT-based vertical plant wall for indoor climate control”. In: *Building and Environment* 183 (2020), p. 107212.

[37] Yichu Xu et al. “Hyperspectral anomaly detection based on machine learning: An overview”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2022).

[38] Nir Nissim et al. “Detection of malicious PDF files and directions for enhancements: A state-of-the art survey”. In: *Computers & Security* 48 (2015), pp. 246–266.

[39] Jinpei Yan, Yong Qi, and Qifan Rao. “Detecting malware with an ensemble method based on deep neural network”. In: *Security and Communication Networks* 2018 (2018).

[40] Paul Giura and Wei Wang. “A context-based detection framework for advanced persistent threats”. In: *2012 International Conference on Cyber Security*. IEEE. 2012, pp. 69–74.

[41] Xu Wang et al. “Detection of command and control in advanced persistent threat based on independent access”. In: *2016 IEEE International Conference on Communications (ICC)*. IEEE. 2016, pp. 1–6.

[42] Guillaume Brogi and Valérie Viet Triem Tong. “Terminaptor: Highlighting advanced persistent threats through information flow tracking”. In: *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE. 2016, pp. 1–5.

[43] Maryam Panahnejad and Meghdad Mirabi. “APT-Dt-KC: advanced persistent threat detection based on kill-chain model”. In: *The Journal of Supercomputing* (2022), pp. 1–34.

[44] Abebe Abeshu Diro and Naveen Chilamkurti. “Distributed attack detection scheme using deep learning approach for Internet of Things”. In: *Future Generation Computer Systems* 82 (2018), pp. 761–768.

[45] Yirui Wu, Dabao Wei, and Jun Feng. “Network attacks detection methods based on deep learning techniques: a survey”. In: *Security and Communication Networks* 2020 (2020), pp. 1–17.

[46] Rajesh Kumar, Siddhant Singh, and Rohan Kela. “Analyzing advanced persistent threats using game theory: A critical literature review”. In: *Critical Infrastructure Protection XV: 15th IFIP WG 11.10 International Conference, ICCIP 2021, Virtual Event, March 15–16, 2021, Revised Selected Papers 15*. Springer. 2022, pp. 45–69.

[47] Md Monowar Anjum, Shahrear Iqbal, and Benoit Hamelin. “ANUBIS: a provenance graph-based framework for advanced persistent threat detection”. In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 2022, pp. 1684–1693.

[48] Hao Zhang et al. “A real-time and ubiquitous network attack detection based on deep belief network and support vector machine”. In: *IEEE/CAA Journal of Automatica Sinica* 7.3 (2020), pp. 790–799.

[49] Rory Coulter et al. “Domain adaptation for Windows advanced persistent threat detection”. In: *Computers & Security* 112 (2022), p. 102496.

[50] Jaafer Al-Saraireh et al. “A novel approach for detecting advanced persistent threats”. In: *Egyptian Informatics Journal* 23.4 (2022), pp. 45–55.

[51] Hope Nkiruka Eke, Andrei Petrovski, and Hatem Ahriz. “The use of machine learning algorithms for detecting advanced persistent threats”. In: *Proceedings of the 12th International Conference on Security of Information and Networks*. 2019, pp. 1–8.

[52] Hope Eke, Andrei Petrovski, and Hatem Ahriz. “Detection of False Command and Response Injection Attacks for Cyber Physical Systems Security and Resilience”. In: *13th International Conference on Security of Information and Networks*. 2020, pp. 1–8.

[53] Thomas H Morris, Zach Thornton, and Ian Turnipseed. “Industrial control system simulation and data logging for intrusion detection system research”. In: *7th annual southeastern cyber security summit* (2015), pp. 3–4.

[54] Xiaoyong Yuan. "Phd forum: Deep learning-based real-time malware detection with multi-stage analysis". In: *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE. 2017, pp. 1–2.

[55] Adel Alshamrani et al. "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities". In: *IEEE Communications Surveys & Tutorials* 21.2 (2019), pp. 1851–1877.

[56] Atif Ahmad et al. "Strategically-motivated advanced persistent threat: Definition, process, tactics and a disinformation model of counterattack". In: *Computers & Security* (2019).

[57] Michal Smiraus and Roman Jasek. "Risks of advanced persistent threats and defense against them". In: *Annals of DAAAM & Proceedings* 1589 (2011).

[58] Command Five. "Advanced Persistent Threats: A Decade in Review". In: *Command Five PTY LTD* (2011), pp. 1–13.

[59] ISACA. "Advanced persistent threat awareness study results: Information systems auditing manual". In: *Information Systems Audit and Control Association* (2014). Accessed on 2019-02-20.

[60] Trend Micro. "Countering the advanced persistent threat challenge with deep discovery". In: *Retrieved* 10.10 (2013), p. 2015.

[61] Jiageng Chen et al. *Special issue on advanced persistent threat*. 2018.

[62] Roger Cressey. *Cyber Security Issues and Challenges*. https://www.aregnet.org/ar/%D8%A7%D9%84%D9%86%D8%B4%D8%A7%D8%B7%D8%A7%D8%AA/item/download/84_61bec66cdcb5bfc4fd6e020da41da46d. Accessed on 2019-02-20. 2012.

[63] Gary Locke Patrick D. Gallagher. *Managing information security risk, organization, mission, and information system view*. https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-39.pdf. 2011.

[64] Olivier Thonnard et al. "Industrial espionage and targeted attacks: Understanding the characteristics of an escalating threat". In: *International workshop on recent advances in intrusion detection*. Springer. 2012, pp. 64–85.

[65] Ping Chen, Lieven Desmet, and Christophe Huygens. "A study on advanced persistent threats". In: *IFIP International Conference on Communications and Multimedia Security*. Springer. 2014, pp. 63–72.

[66] Mandiant Intelligence Center. "APT1: Exposing one of China's cyber espionage units". In: *Mandian. com* (2013).

[67] Symantec. "Energy/Oil and Gas Report: Protecting critical systems while promoting operational efficiency - towards the digital oilfield, Internet Security Threat Report (ISTR)". In: 17 (2012). Accessed on 2018-02-17.

[68] Dmitri Alperovitch et al. *Revealed: operation shady RAT*. Vol. 3. McAfee, 2011.

[69] McAfee Foundstone Professional Services - Firm. *Global Energy Cyberattacks:" Night Dragon"*. McAfee, Incorporated, 2011.

[70] Stuart McClure et al. "Protecting your critical assets-lessons learned from operation aurora". In: *Tech. Rep.* (2010).

[71] Murtaza A Siddiqi and Naveed Ghani. "Critical Analysis on Advanced Persistent Threats". In: *Int. J. Comput. Appl.* 141.13 (2016), pp. 46–50.

[72] Murray Brand, Craig Valli, and Andrew Woodward. "Malware Forensics: Discovery of the intent of Deception". In: *Journal of Digital Forensics, Security and Law* 5.4 (2010), p. 2.

[73] Narasimha Shashidhar and Lei Chen. "A phishing model and its applications to evaluating phishing attacks". In: (2011).

[74] Aaron Zimba et al. “Modeling and detection of the multi-stages of Advanced Persistent Threats attacks based on semi-supervised learning and complex networks characteristics”. In: *Future Generation Computer Systems* 106 (2020), pp. 501–517.

[75] Nart Villeneuve et al. “Operation Ke3chang: Targeted Attacks Against Ministries of Foreign Affairs”. In: *Fire Eye* (2013).

[76] Saurabh Singh et al. “A comprehensive study on APT attacks and countermeasures for future networks and communications: challenges and solutions”. In: *The Journal of Supercomputing* (2016), pp. 1–32.

[77] Eric M Hutchins, Michael J Cloppert, Rohan M Amin, et al. “Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains”. In: *Leading Issues in Information Warfare & Security Research* 1.1 (2011), p. 80.

[78] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. “An effective address mutation approach for disrupting reconnaissance attacks”. In: *IEEE Transactions on Information Forensics and Security* 10.12 (2015), pp. 2562–2577.

[79] Hamza Saleem and Muhammad Naveed. “SoK: Anatomy of data breaches.” In: *Proc. Priv. Enhancing Technol.* 2020.4 (2020), pp. 153–174.

[80] Zuoguang Wang, Hongsong Zhu, and Limin Sun. “Social engineering in cybersecurity: Effect mechanisms, human vulnerabilities and attack methods”. In: *IEEE Access* 9 (2021), pp. 11895–11910.

[81] Tian Lin et al. “Susceptibility to spear-phishing emails: Effects of internet user demographics and email content”. In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 26.5 (2019), pp. 1–28.

[82] Benjamin Reinheimer et al. “An investigation of phishing awareness and education over time: When and how to best remind users”. In: *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. 2020, pp. 259–284.

[83] Xiaoqing Sun and Jiahai Yang. “HetGLM: Lateral Movement Detection by Discovering Anomalous Links with Heterogeneous Graph Neural Network”. In: *2022 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE. 2022, pp. 404–411.

[84] Ahmad O Almashhadani et al. “MaldomDetector: A system for detecting algorithmically generated domain names with machine learning”. In: *Computers & Security* 93 (2020), p. 101787.

[85] Ishai Rosenberg et al. “Adversarial machine learning attacks and defense methods in the cyber security domain”. In: *ACM Computing Surveys (CSUR)* 54.5 (2021), pp. 1–36.

[86] Hatma Suryotrisongko et al. “Robust botnet DGA detection: Blending XAI and OSINT for cyber threat intelligence sharing”. In: *IEEE Access* 10 (2022), pp. 34613–34624.

[87] Yi Li et al. “A machine learning framework for domain generation algorithm-based malware detection”. In: *IEEE Access* 7 (2019), pp. 32765–32782.

[88] Akhila GP and Angelin Gladston. “A machine learning framework for domain generating algorithm based malware detection”. In: *Security and Privacy* 3.6 (2020), e127.

[89] Yang Xin et al. “Machine learning and deep learning methods for cybersecurity”. In: *Ieee access* 6 (2018), pp. 35365–35381.

[90] Ishan Karunanayake et al. “De-anonymisation attacks on Tor: A Survey”. In: *IEEE Communications Surveys & Tutorials* 23.4 (2021), pp. 2324–2350.

[91] Debajyoti Das et al. “Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency-choose two”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 108–126.

[92] Norbert Tihanyi et al. “Privacy-Preserving Password Cracking: How a Third Party Can Crack Our Password Hash Without Learning the Hash Value or the Cleartext”. In: *arXiv preprint arXiv:2306.08740* (2023).

[93] Kutub Thakur et al. “Impact of cyber-attacks on critical infrastructure”. In: *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE. 2016, pp. 183–186.

[94] NJCCIC. “CRASHOVERRIDE NJCCIC Threat Profile, official site of the state of new jersey Original Release Date: 2017-08-10 and accessed on 16/07/20”. In: *NJCCIC* (2017). URL: https://www.cyber.nj.gov/threat-center/threat-profiles/ics-malware-variants/crashoverride.

[95] Joseph Slowik. “Evolution of ICS Attacks and the Prospects for Future Disruptive Events”. In: *Threat Intelligence Centre Dragos Inc* (2019).

[96] Robert M Lee, MJ Assante, and T Conway. “CRASHOVERRIDE: Analysis of the threat to electric grid operations”. In: *Dragos Inc., March* (2017).

[97] Kevin E Hemsley, E Fisher, et al. *History of industrial control system cyber incidents*. Tech. rep. Idaho National Lab.(INL), Idaho Falls, ID (United States), 2018.

[98] MITRE ATT&CK. “Deep Panda”. In: (2021). URL: https://attack.mitre.org/groups/G0009/.

[99] Umara Noor et al. “A machine learning-based FinTech cyber threat attribution framework using high-level indicators of compromise”. In: *Future Generation Computer Systems* 96 (2019), pp. 227–242.

[100] Kaspersky. “The Epic Turla (snake/Uroburos) attacks”. In: (2021). URL: https://www.kaspersky.co.uk/resource-center/threats/epic-turla-snake-malware-attacks.

[101] Javad Hassannataj Joloudari et al. “Early detection of the advanced persistent threat attack using performance analysis of deep learning”. In: *IEEE Access* 8 (2020), pp. 186125–186137.

[102] Peter E Kaloroumakis and Michael J Smith. “Toward a knowledge graph of cybersecurity countermeasures”. In: *The MITRE Corporation* 11 (2021).

[103] Best Practical Solutions. “Best-Practical-Solutions: Request tracker (RT)”. In: (2015). Accessed on 2018-12-02. URL: https://bestpractical.com/request-tracker.

[104] Blake E Strom et al. “Finding cyber threats with ATT&CK-based analytics”. In: *The MITRE Corporation, Bedford, MA, Technical Report No. MTR170202* (2017).

[105] Critical Infrastructure Cybersecurity. “Framework for improving critical infrastructure cybersecurity”. In: *URL: https://nvlpubs. nist. gov/nistpubs/CSWP/NIST. CSWP* 4162018 (2018).

[106] Ron Ross-NIST. “NIST Special Publication 800-53, Revision 4, Security and Privacy Controls for Federal Information Systems and Organizations”. In: (2013).

[107] Peter Clay. “A modern threat response framework”. In: *Network Security* 2015.4 (2015), pp. 5–10.

[108] Mathew Nicho, Adelaiye Oluwasegun, and Faouzi Kamoun. “Identifying vulnerabilities in apt attacks: A simulated approach”. In: *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE. 2018, pp. 1–4.

[109] Zakiah Zulkefli, Manmeet Mahinderjit Singh, and Nurul Hashimah Ahamed Hassain Malim. “Advanced persistent threat mitigation using multi level security–access control framework”. In: *International Conference on Computational Science and Its Applications*. Springer. 2015, pp. 90–105.

[110] J Vukalović and Damir Delija. “Advanced persistent threats-detection and defense”. In: *2015 38Th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE. 2015, pp. 1324–1330.

[111] Hope Nkiruka Eke. *"Study of Web-based Communities: Study of the structure of online communities and application of statistical methods to evaluate users types & behaviours"*. LAP LAMBERT Academic, 2016, pp. 1–168.

[112] Ghita Berrada et al. “A baseline for unsupervised advanced persistent threat detection in system-level provenance”. In: *Future Generation Computer Systems* 108 (2020), pp. 401–413.

[113] Saranya Chandran, P Hrudya, and Prabaharan Poornachandran. “An efficient classification model for detecting advanced persistent threat”. In: *2015 international conference on advances in computing, communications and informatics (ICACCI)*. IEEE. 2015, pp. 2001–2009.

[114] Pengfei Hu et al. “Dynamic defense strategy against advanced persistent threat with insiders”. In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE. 2015, pp. 747–755.

[115] Kun Lv, Yun Chen, and Changzhen Hu. “Dynamic Defense Strategy Against Advanced Persistent Threat Under Heterogeneous Networks”. In: *Information Fusion* (2019).

[116] Pengdeng Li et al. “Defending against the advanced persistent threat: An optimal control approach”. In: *Security and Communication Networks* 2018 (2018).

[117] Accenture. “The Journey to Sustainable NERC CIP Compliance, Accenture”. In: (2018).

[118] Eric Ke Wang et al. “Security issues and challenges for cyber physical system”. In: *2010 IEEE/ACM Int’l Conference on Green Computing and Communications & Int’l Conference on Cyber, Physical and Social Computing*. IEEE. 2010, pp. 733–738.

[119] Wenye Wang and Zhuo Lu. “Cyber security in the smart grid: Survey and challenges”. In: *Computer networks* 57.5 (2013), pp. 1344–1371.

[120] Chakib Bekara. “Security issues and challenges for the IoT-based smart grid.” In: *FNC/MobiSPC*. 2014, pp. 532–537.

[121] Soman KP, Mamoun Alazab, et al. “A Comprehensive Tutorial and Survey of Applications of Deep Learning for Cyber Security”. In: (2020).

[122] Artem A Nazarenko and Ghazanfar Ali Safdar. “Survey on security and privacy issues in cyber physical systems [J]”. In: *AIMS Electronics and Electrical Engineering* 3.2 (2019), pp. 111–143.

[123] Markus Brändle and Martin Naedele. “Security for process control systems: An overview”. In: *IEEE Security & Privacy* 6.6 (2008), pp. 24–29.

[124] Jason Madden, Bruce McMillin, and Anik Sinha. “Environmental obfuscation of a cyber physical system-vehicle example”. In: *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*. IEEE. 2010, pp. 176–181.

[125] Mona Nasser Almansoori, Ahmad Ahmad Elshamy, and Ahmad Abdel Muttalib Mustafa. “Secure z-mac protocol as a proposed solution for improving security in wsns”. In: *Information* 13.3 (2022), p. 105.

[126] Myriam Dunn Cavelty. “Cyber-security”. In: *The routledge handbook of new security studies*. Routledge, 2010, pp. 166–174.

[127] Wen Yang and Qianchuan Zhao. “Cyber security issues of critical components for industrial control system”. In: *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*. IEEE. 2014, pp. 2698–2703.

[128] W Bolton. *Programmable Logic Controllers, Newnes*. 2009.

[129] Morten Gjendemsjø. “Security in Programmable Logic Controllers”. In: (2012).

[130] Lorena Cazorla, Cristina Alcaraz, and Javier Lopez. “Cyber stealth attacks in critical information infrastructures”. In: *IEEE Systems Journal* 12.2 (2016), pp. 1778–1792.

[131] USDHS ICS-CERT. “ICS-monitor incident response activity; Vulnerability Coordination”. In: *National Cybersecurity and Communications Integration Center* (2014).

[132] Markus Jakobsson, Susanne Wetzel, and Bülent Yener. “Stealth attacks on ad-hoc wireless networks”. In: *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No. 03CH37484)*. Vol. 3. IEEE. 2003, pp. 2103–2111.

[133] Vera Marinova-Boncheva. “A short survey of intrusion detection systems”. In: *problems of Engineering Cybernetics and Robotics* 58 (2007), pp. 23–30.

[134] G Joy Persial, M Prabhu, and R Shanmugalakshmi. “Side channel attack-survey”. In: *Int. J. Adv. Sci. Res. Rev* 1.4 (2011), pp. 54–57.

[135] Richard J Barnett and Barry Irwin. “Towards a taxonomy of network scanning techniques”. In: *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*. 2008, pp. 1–7.

[136] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich. “Internet intrusions: Global characteristics and prevalence”. In: *ACM SIGMETRICS Performance Evaluation Review* 31.1 (2003), pp. 138–147.

[137] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. “Cyber scanning: a comprehensive survey”. In: *Ieee communications surveys & tutorials* 16.3 (2013), pp. 1496–1519.

[138] Hung Nguyen Viet et al. “Using deep learning model for network scanning detection”. In: *Proceedings of the 4th International Conference on Frontiers of Educational Technologies*. 2018, pp. 117–121.

[139] Marco De Vivo et al. “A review of port scanning techniques”. In: *ACM SIGCOMM Computer Communication Review* 29.2 (1999), pp. 41–48.

[140] Chunmei Yin et al. “Honeypot and scan detection in intrusion detection system”. In: *Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No. 04CH37513)*. Vol. 2. IEEE. 2004, pp. 1107–1110.

[141] Jingfei Kong et al. “Hardware-software integrated approaches to defend against software cache-based side channel attacks”. In: *2009 IEEE 15th International Symposium on High Performance Computer Architecture*. IEEE. 2009, pp. 393–404.

[142] Nishi Tomar and Manoj Singh Gaur. “Information theft through covert channel by exploiting HTTP post method”. In: *2013 Tenth International Conference on Wireless and Optical Communications Networks (WOCN)*. IEEE. 2013, pp. 1–5.

[143] Butler W Lampson. “A note on the confinement problem”. In: *Communications of the ACM* 16.10 (1973), pp. 613–615.

[144] Xing Jin et al. “Code injection attacks on html5-based mobile apps: Characterization, detection and mitigation”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 2014, pp. 66–77.

[145] Anjee Gorkhali and Li Da Xu. “Enterprise architecture: a literature review”. In: *Journal of Industrial Integration and Management* 2.02 (2017), p. 1750009.

[146] David Romero et al. “Towards a human-centred reference architecture for next generation balanced automation systems: human-automation symbiosis”. In: *IFIP international conference on advances in production management systems*. Springer. 2015, pp. 556–566.

[147] Hong Li and Theodore J Williams. “A Vision of Enterprise Integration Considerations”. In: *Knowledge Sharing in the Integrated Enterprise*. Springer, 2004, pp. 249–267.

[148] Blake E Strom et al. “Mitre att&ck: Design and philosophy”. In: *Technical report* (2018).

[149] Przemysław Buczkowski et al. “Optimal Security Hardening over a Probabilistic Attack Graph: A Case Study of an Industrial Control System using CySecTool”. In: *Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*. 2022, pp. 21–30.

[150] Vladimir N Kozlovsky et al. “Determination of the causes of the excess of the level of electromagnetic interference from the ignition system using an intelligent diagnostic system”. In: *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE. 2019, pp. 569–573.

[151] Theodore J Williams. “The Purdue enterprise reference architecture”. In: *IFAC Proceedings Volumes* 26.2 (1993), pp. 559–564.

[152] Hong Li and Theodore J Williams. “Interface design for the Purdue enterprise reference architecture (PERA) and methodology in e-Work”. In: *Production Planning & Control* 14.8 (2003), pp. 704–719.

[153] H Li and TJ Williams. “A formalization and extension of the Purdue Enterprise Reference Architecture and the Purdue Methodology (Report 158)”. In: *Purdue Laboratory for Applied Industrial Control, West Lafayette, Indiana* (1994).

[154] Rawan Al-Shaer, Jonathan M Spring, and Eliana Christou. “Learning the Associations of MITRE ATT&CK Adversarial Techniques”. In: *arXiv preprint arXiv:2005.01654* (2020).

[155] MITRE. “MITRE ATT&CK® Matrix for Enterprise”. In: (2019). URL: https://attack.mitre.org/matrices/enterprise.

[156] Anomali. “What Is MITRE ATT&CK and How Is It Useful”. In: (2020). URL: https://www.anomali.com/resources/what-mitre-attck-is-and-how-it-is-useful.

[157] Karen Scarfone, Peter Mell, et al. “Guide to intrusion detection and prevention systems (idps)”. In: *NIST special publication* 800.2007 (2007), p. 94.

[158] Ahmed Patel, Qais Qassim, and Christopher Wills. “A survey of intrusion detection and prevention systems”. In: *Information Management & Computer Security* (2010).

[159] Soubhik Das and Manisha J Nene. “A survey on types of machine learning techniques in intrusion prevention systems”. In: *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*. IEEE. 2017, pp. 2296–2299.

[160] C Martin. *What is IPS and how intrusion prevention system works*. 2009.

[161] David Mudzingwa and Rajeev Agrawal. “A study of methodologies used in intrusion detection and prevention systems (IDPS)”. In: *2012 Proceedings of IEEE Southeastcon*. IEEE. 2012, pp. 1–6.

[162] Panagiotis I Radoglou-Grammatikis and Panagiotis G Sarigiannidis. “Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems”. In: *IEEE Access* 7 (2019), pp. 46595–46620.

[163] Hung-Jen Liao et al. “Intrusion detection system: A comprehensive review”. In: *Journal of Network and Computer Applications* 36.1 (2013), pp. 16–24.

[164] Arun Viswanathan, Kymie Tan, and Clifford Neuman. “Deconstructing the assessment of anomaly-based intrusion detectors”. In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2013, pp. 286–306.

[165] Dorothy E Denning. “An intrusion-detection model”. In: *IEEE Transactions on software engineering* 2 (1987), pp. 222–232.

[166] Indraneel Mukhopadhyay, Mohuya Chakraborty, Satyajit Chakrabarti, et al. “A comparative study of related technologies of intrusion detection & prevention systems”. In: *Journal of Information Security* 2.01 (2011), p. 28.

[167] Justin Lee, Stuart Moskovics, and Lucas Silacci. “A survey of intrusion detection analysis methods”. In: *University of California, san Diego* (1999).

[168] Ross Heenan and Naghmeh Moradpoor. “A survey of Intrusion Detection System technologies”. In: *The First Post Graduate Cyber Security Symposium*. 2016.

[169] Ahmed Patel et al. “An intrusion detection and prevention system in cloud computing: A systematic review”. In: *Journal of network and computer applications* 36.1 (2013), pp. 25–41.

[170] Andreas Fuchsberger. “Intrusion detection systems and intrusion prevention systems”. In: *Information Security Technical Report* 10.3 (2005), pp. 134–139.

[171] Geoffrey E Hinton and Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *science* 313.5786 (2006), pp. 504–507.

[172] Rahul Katarya and Shubham Rastogi. “A study on neural networks approach to time-series analysis”. In: *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. IEEE. 2018, pp. 116–119.

[173] Emma Hart. “Towards Lifelong Learning in Optimisation Algorithms”. In: *Proceedings of the 9th International Joint Conference on Computational Intelligence*. Vol. 1. 2017, pp. 7–9.

[174] Yoshua Bengio, Yann LeCun, et al. “Scaling learning algorithms towards AI”. In: *Large-scale kernel machines* 34.5 (2007), pp. 1–41.

[175] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. “Deep neural networks for object detection”. In: (2013).

[176] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[177] Ahmed Alghamdi et al. “Detection of myocardial infarction based on novel deep transfer learning methods for urban healthcare in smart cities”. In: *Multimedia tools and applications* (2020), pp. 1–22.

[178] Ferhat Ozgur Catak and Ahmet Fatih Mustacoglu. “Distributed denial of service attack detection using autoencoder and deep neural networks”. In: *Journal of Intelligent & Fuzzy Systems* 37.3 (2019), pp. 3969–3979.

[179] JQ James, Yunhe Hou, and Victor OK Li. “Online false data injection attack detection with wavelet transform and deep neural networks”. In: *IEEE Transactions on Industrial Informatics* 14.7 (2018), pp. 3271–3280.

[180] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. “Evaluation of recurrent neural network and its variants for intrusion detection system (IDS)”. In: *International Journal of Information System Modeling and Design (IJISMD)* 8.3 (2017), pp. 43–63.

[181] Ian P Turnipseed. *A new scada dataset for intrusion detection research*. Mississippi State University, 2015.

[182] Nour Moustafa. “Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic.” PhD thesis. University of New South Wales, Canberra, Australia, 2017.

[183] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. “Methods for interpreting and understanding deep neural networks”. In: *Digital Signal Processing* 73 (2018), pp. 1–15.

[184] Paul J Werbos. “Applications of advances in nonlinear sensitivity analysis”. In: *System modeling and optimization*. Springer, 1982, pp. 762–770.

[185] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.

[186] Mohammed Hasan Ali and Mohamad Fadli Zolkipli. “Review on hybrid extreme learning machine and genetic algorithm to work as intrusion detection system in cloud computing”. In: *vol* 11 (2016), pp. 460–464.

[187] Geoffrey E Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).

[188] Soman KP et al. “RNNSecureNet: Recurrent neural networks for Cyber security use-cases”. In: *arXiv preprint arXiv:1901.04281* (2019).

[189] Y LeCun, Y Bengio, and G Hinton. “Deep learning. nature 521 (7553): 436”. In: *Google Scholar* (2015).

[190] Alex Graves. “Supervised sequence labelling”. In: *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 5–13.

[191] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[192] Jie Ling et al. “An intrusion detection method for industrial control systems based on bidirectional simple recurrent unit”. In: *Computers & Electrical Engineering* 91 (2021), p. 107049.

[193] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[194] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[195] T Mikolov et al. “Recurrent Neural Network Based Language Model.[in:] 11th Annual Conference of the International Speech Communication Association”. In: Interspeech. 2010.

[196] Stefan Kombrink et al. “Recurrent neural network based language modeling in meeting recognition”. In: *Twelfth annual conference of the international speech communication association*. 2011.

[197] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[198] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[199] Ilya Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, Canada, 2013.

[200] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[201] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. PMLR. 2013, pp. 1310–1318.

[202] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[203] Alex Graves and Jürgen Schmidhuber. “Framewise phoneme classification with bidirectional LSTM and other neural network architectures”. In: *Neural networks* 18.5-6 (2005), pp. 602–610.

[204] Alex Graves. “Generating sequences with recurrent neural networks”. In: *arXiv preprint arXiv:1308.0850* (2013).

[205] Hermann Mayer et al. “A system for robotic heart surgery that learns to tie knots using recurrent neural networks”. In: *Advanced Robotics* 22.13-14 (2008), pp. 1521–1537.

[206] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. Ieee. 2013, pp. 6645–6649.

[207] Andrew Maas et al. "Recurrent neural networks for noise reduction in robust ASR". In: (2012).

[208] Marek Ostaszewski, Franciszek Seredynski, and Pascal Bouvry. "Immune anomaly detection enhanced with evolutionary paradigms". In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. 2006, pp. 119–126.

[209] Zachary C Lipton, John Berkowitz, and Charles Elkan. "A critical review of recurrent neural networks for sequence learning". In: *arXiv preprint arXiv:1506.00019* (2015).

[210] Razvan Pascanu et al. "How to construct deep recurrent neural networks". In: *arXiv preprint arXiv:1312.6026* (2013).

[211] Matthew G Schultz et al. "Data mining methods for detection of new malicious executables". In: *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*. IEEE. 2000, pp. 38–49.

[212] Nidhi Singh et al. "Sentiment score analysis and topic modelling For GST implementation in India". In: *Soft Computing for Problem Solving*. Springer, 2019, pp. 243–254.

[213] Yan-Yan Song and LU Ying. "Decision tree methods: applications for classification and prediction". In: *Shanghai archives of psychiatry* 27.2 (2015), p. 130.

[214] Tarek Lajnef et al. "Learning machines and sleeping brains: automatic sleep stage classification using decision-tree multi-class support vector machines". In: *Journal of neuroscience methods* 250 (2015), pp. 94–105.

[215] Glenn Carl et al. "Denial-of-service attack-detection techniques". In: *IEEE Internet computing* 10.1 (2006), pp. 82–89.

[216] Manar Abu Talib et al. "APT beaconing detection: A systematic review". In: *Computers & Security* (2022), p. 102875.

[217] Siwar Kriaa, Marc Bouissou, and Ludovic Piètre-Cambacédès. "Modeling the Stuxnet attack with BDMP: Towards more formal risk assessments". In: *2012 7th International Conference on Risks and Security of Internet and Systems (CRiSIS)*. IEEE. 2012, pp. 1–8.

[218] Michael J. Assante and Robert M. Lee. "The Industrial Control System Cyber Kill Chain". In: (2015).

[219] Mark Kedgley. "If you can't stop the breach, at least spot the breach". In: *Network Security* 2015.4 (2015), pp. 11–12.

[220] Ondrej Linda, Todd Vollmer, and Milos Manic. "Neural network based intrusion detection system for critical infrastructures". In: *2009 international joint conference on neural networks*. IEEE. 2009, pp. 1827–1834.

[221] Mirco Marchetti et al. "Analysis of high volumes of network traffic for advanced persistent threat detection". In: *Computer Networks* 109 (2016), pp. 127–141.

[222] Daesung Moon et al. "DTB-IDS: an intrusion detection system based on decision tree using behavior analysis for preventing APT attacks". In: *The Journal of supercomputing* 73.7 (2017), pp. 2881–2895.

[223] Jafar Haadi Jafarian, Masoumeh Abolfathi, and Mahsa Rahimian. "Detecting Network Scanning Through Monitoring and Manipulation of DNS Traffic". In: *IEEE Access* 11 (2023), pp. 20267–20283.

[224] Marco Balduzzi, Vincenzo Ciangaglini, and Robert McArdle. "Targeted attacks detection with spunge". In: *2013 Eleventh Annual Conference on Privacy, Security and Trust*. IEEE. 2013, pp. 185–194.

[225] Vincenzo Ciangaglini Marco Balduzzi and Robert McArdle. "Targeted attacks detection with SPuNge". In: (2013). Accessed on 2019-03-10.

[226] McMillan Scott et al. *C3E DNS Challenge Report*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA PITTSBURGH United States, 2013.

[227] Paul S Ferrell. *Apt infection discovery using DNS data*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2013.

[228] Parth Bhatt, Edgar Toshiro Yano, and Per Gustavsson. "Towards a framework to detect multi-stage advanced persistent threats attacks". In: *2014 IEEE 8th International Symposium on Service Oriented System Engineering*. IEEE. 2014, pp. 390–395.

[229] Ibrahim Ghafir et al. "Detection of advanced persistent threat using machine-learning correlation analysis". In: *Future Generation Computer Systems* 89 (2018), pp. 349–359.

[230] Georgios Ioannou et al. "A Markov multi-phase transferable belief model: An application for predicting data exfiltration APTs". In: *Proceedings of the 16th International Conference on Information Fusion*. IEEE. 2013, pp. 842–849.

[231] Ibrahim Ghafir et al. "Hidden Markov models and alert correlations for the prediction of advanced persistent threats". In: *IEEE Access* 7 (2019), pp. 99508–99520.

[232] APT_Alert_dataset.db. "Dataset of Advanced Persistent Threat (APT) alerts". In: (Jan. 2019). DOI: 10.17028/rd.lboro.7577750.v1. URL: https://repository.lboro.ac.uk/articles/dataset/Dataset_of_Advanced_Persistent_Threat_APT_alerts/7577750/1.

[233] Nedim Šrndic and Pavel Laskov. "Detection of malicious pdf files based on hierarchical document structure". In: *Proceedings of the 20th Annual Network & Distributed System Security Symposium*. 2013, pp. 1–16.

[234] Ibrahim Ghafir, Mohammad Hammoudeh, and Vaclav Prenosil. "Defending against the advanced persistent threat Detection of disguised executable files". In: *PeerJ Prepr* 6 (2018), e2998.

[235] Rory Coulter et al. "Domain Adaptation for Windows Advanced Persistent Threat Detection". In: *Computers & Security* (2021), p. 102496.

[236] Do Xuan Cho and Ha Hai Nam. "A method of monitoring and detecting APT attacks based on unknown domains". In: *Procedia Computer Science* 150 (2019), pp. 316–323.

[237] Cho Do Xuan, Mai Hoang Dao, and Hoa Dinh Nguyen. "APT attack detection based on flow network analysis techniques using deep learning". In: *Journal of Intelligent & Fuzzy Systems* 39.3 (2020), pp. 4785–4801.

[238] Abdelrahman Ayad, Mohsen Khalaf, and Ehab El-Saadany. "Detection of false data injection attacks in automatic generation control systems considering system nonlinearities". In: *2018 IEEE Electrical Power and Energy Conference (EPEC)*. IEEE. 2018, pp. 1–6.

[239] Ru Zhang et al. "Construction of two statistical anomaly features for small-sample apt attack traffic classification". In: *arXiv preprint arXiv:2010.13978* (2020).

[240] Prasanta Gogoi et al. "Packet and flow based network intrusion dataset". In: *International Conference on Contemporary Computing*. Springer. 2012, pp. 322–334.

[241] Branka Stojanović, Katharina Hofer-Schmitz, and Ulrike Kleb. "APT datasets and attack modeling for automated detection methods: A review". In: *Computers & Security* 92 (2020), p. 101734.

[242] José A Sáez, Bartosz Krawczyk, and Michał Woźniak. "Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets". In: *Pattern Recognition* 57 (2016), pp. 164–178.

[243] Pattaramon Vuttipittayamongkol and Eyad Elyan. "Improved overlap-based undersampling for imbalanced dataset classification with application to epilepsy and parkinson's disease". In: *International journal of neural systems* 30.08 (2020), p. 2050043.

[244] Cho Do Xuan and Mai Hoang Dao. "A novel approach for APT attack detection based on combined deep learning model". In: *Neural Computing and Applications* 33.20 (2021), pp. 13251–13264.

[245] Johan Sigholm and Martin Bang. "Towards offensive cyber counterintelligence: Adopting a target-centric view on advanced persistent threats". In: *2013 European Intelligence and Security Informatics Conference*. IEEE. 2013, pp. 166–171.

[246] Joseph Sexton, Curtis Storlie, and Joshua Neil. "Attack chain detection". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 8.5-6 (2015), pp. 353–363.

[247] J Vijaya Chandra, Narasimham Challa, and Sai Kiran Pasupuleti. "A practical approach to E-mail spam filters to protect data from advanced persistent threat". In: *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*. IEEE. 2016, pp. 1–5.

[248] Sadegh Momeni Milajerdi and Mehdi Kharrazi. "A Composite-Metric Based Path Selection Technique for the Tor Anonymity Network". In: *arXiv e-prints* (2017), arXiv–1707.

[249] Stephan Alexander Kollmann. "Privacy-preserving decentralised collaborative applications". PhD thesis. University of Cambridge, 2019.

[250] Mehran Alidoost Nia and A Ruiz-Martínez. "Analytical survey of existing research on anonymous communication technologies and directions for further study". In: *International Journal of Management Research and Reviews* 12.4 (2022), pp. 13–33.

[251] Cho Do Xuan, Dung Kim Nguyen, and Duc Tran Duong. "A multi-layer approach for advanced persistent threat detection using machine learning based on network traffic". In: *Journal of Intelligent & Fuzzy Systems* Preprint (), pp. 1–19.

[252] Thomas Morris and Wei Gao. "Industrial control system traffic data sets for intrusion detection research". In: *International Conference on Critical Infrastructure Protection*. Springer. 2014, pp. 65–78.

[253] Nour Moustafa and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)". In: *2015 military communications and information systems conference (MilCIS)*. IEEE. 2015, pp. 1–6.

[254] L Dhanabal and SP Shantharajah. "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms". In: *International journal of advanced research in computer and communication engineering* 4.6 (2015), pp. 446–452.

[255] Mahbod Tavallaee et al. "A detailed analysis of the KDD CUP 99 data set". In: *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE. 2009, pp. 1–6.

[256] N Moustaf and Jill Slay. "Creating novel features to anomaly network detection using DARPA-2009 data set". In: *Proceedings of the 14th European Conference on Cyber Warfare and Security. Academic Conferences Limited*. 2015, pp. 204–212.

[257] John McHugh. "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory". In: *ACM Transactions on Information and System Security (TISSEC)* 3.4 (2000), pp. 262–294.

[258] ARi Vasudevan, E Harshini, and S Selvakumar. "SSENet-2011: a network intrusion detection system dataset and its comparison with KDD CUP 99 dataset". In: *2011 second asian himalayas international conference on internet (AH-ICI)*. IEEE. 2011, pp. 1–5.

[259] Matthew V Mahoney and Philip K Chan. "An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection". In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2003, pp. 220–237.

[260] Tavish Vaidya. “2001-2013: Survey and Analysis of Major Cyberattacks”. In: *arXiv preprint arXiv:1507.06673* (2015).

[261] BJ Bejoy and S Janakiraman. “Artificial immune system based intrusion detection systems—a comprehensive review”. In: *Int J Comput Eng Technol* 8.1 (2017), pp. 85–95.

[262] Bin Jia, Zhaowen Lin, and Yan Ma. “Advanced Persistent Threat Detection Method Research Based on Relevant Algorithms to Artificial Immune System”. In: *International Conference on Trustworthy Computing and Services*. Springer. 2014, pp. 221–228.

[263] Christopher D McDermott, Farzan Majdani, and Andrei V Petrovski. “Botnet detection in the internet of things using deep learning approaches”. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–8.

[264] Ralf C Staudemeyer. “Applying long short-term memory recurrent neural networks to intrusion detection”. In: *South African Computer Journal* 56.1 (2015), pp. 136–154.

[265] Lalit K Mestha, Olugbenga M Anubi, and Masoud Abbaszadeh. “Cyber-attack detection and accommodation algorithm for energy delivery systems”. In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE. 2017, pp. 1326–1331.

[266] S Venkatesan et al. “Artificial immune system based mobile agent platform protection”. In: *Computer Standards & Interfaces* 35.4 (2013), pp. 365–373.

[267] Kevin Sim, Emma Hart, and Ben Paechter. “A lifelong learning hyper-heuristic method for bin packing”. In: *Evolutionary computation* 23.1 (2015), pp. 37–67.

[268] Wanli Ma, Dat Tran, and Dharmendra Sharma. “Negative selection with antigen feedback in intrusion detection”. In: *International Conference on Artificial Immune Systems*. Springer. 2008, pp. 200–209.

[269] Thomas Stibor, Jonathan Timmis, and Claudia Eckert. “A comparative study of real-valued negative selection to statistical anomaly detection techniques”. In: *International Conference on Artificial Immune Systems*. Springer. 2005, pp. 262–275.

[270] Bartosz Krawczyk. “Cost-sensitive one-vs-one ensemble for multi-class imbalanced data”. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 2447–2452.

[271] Zhi-Hua Zhou and Xu-Ying Liu. “Training cost-sensitive neural networks with methods addressing the class imbalance problem”. In: *IEEE Transactions on knowledge and data engineering* 18.1 (2005), pp. 63–77.

[272] Tien Thanh Nguyen et al. “A weighted multiple classifier framework based on random projection”. In: *Information Sciences* 490 (2019), pp. 36–58.

[273] Tien Thanh Nguyen et al. “Combining heterogeneous classifiers via granular prototypes”. In: *Applied Soft Computing* 73 (2018), pp. 795–815.

[274] Bartosz Krawczyk. “Combining one-vs-one decomposition and ensemble learning for multi-class imbalanced data”. In: *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015*. Springer. 2016, pp. 27–36.

[275] Gary M Weiss. “Mining with rarity: a unifying framework”. In: *ACM Sigkdd Explorations Newsletter* 6.1 (2004), pp. 7–19.

[276] Guo Haixiang et al. “Learning from class-imbalanced data: Review of methods and applications”. In: *Expert Systems with Applications* 73 (2017), pp. 220–239.

[277] Andrea Dal Pozzolo et al. “Learned lessons in credit card fraud detection from a practitioner perspective”. In: *Expert systems with applications* 41.10 (2014), pp. 4915–4928.

[278] Claire Cardie and Nicholas Howe. “Improving minority class prediction using case-specific feature weights”. In: (1997).

[279] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. "Classification of imbalanced data: A review". In: *International journal of pattern recognition and artificial intelligence* 23.04 (2009), pp. 687–719.

[280] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD explorations newsletter* 6.1 (2004), pp. 20–29.

[281] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. "A systematic study of the class imbalance problem in convolutional neural networks". In: *Neural networks* 106 (2018), pp. 249–259.

[282] Alberto Fernández et al. "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary". In: *Journal of artificial intelligence research* 61 (2018), pp. 863–905.

[283] Amalia Luque et al. "The impact of class imbalance in classification performance metrics based on the binary confusion matrix". In: *Pattern Recognition* 91 (2019), pp. 216–231.

[284] Esraa Alomari et al. "A survey of botnet-based ddos flooding attacks of application layer: Detection and mitigation approaches". In: *Handbook of research on modern cryptographic solutions for computer and cyber security*. IGI Global, 2016, pp. 52–79.

[285] MGeorge Kurtz. "2023 Global Threat Report." In: (2023). URL: https://go.crowdstrike.com/rs/281-OBQ-266/images/CrowdStrike2023GlobalThreatReport.pdf.%20Accessed%20on%2003%20July%202023.

[286] Vanson Bourne. "Time To Adapt: The Rise Of Advanced Persistent Threats". In: (2023). URL: https://info.gatewatcher.com/en/ty-2023-advanced-persistent-threat-european-study-2023?submissionGuid=ac6e6b65-bffc-4bb8-8e2d-5d81d2ce7be3.%20Accessed%20on%2003%20July%202023..

[287] Ke Li, Chaoge Liu, and Xiang Cui. "Poster: A lightweight unknown http botnets detecting and characterizing system". In: *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 2014, pp. 1454–1456.

[288] Futai Zou et al. "Detecting malware based on DNS graph mining". In: *International Journal of Distributed Sensor Networks* 11.10 (2015), p. 102687.

[289] Abuse.ch. "Abuse.ch. - SSL Blacklist". In: (2023). URL: https://sslbl.abuse.ch/.%20Accessed%20on%2028%20June%202023..

[290] Pierluigi Paganini. "A Security Researcher at Abuse.ch has started SSL blacklist project to create an archive of all the digital certificates used for illicit activities." In: (2014). URL: https://securityaffairs.co/26672/cyber-crime/ssl-blacklist-new-weapon-fight-malware-botnet.html.%20Accessed%20on%2028%20June%202023.

[291] Malware domain list. "Malware domain list." In: (2023). URL: https://www.malwaredomainlist.com/.%20Accessed%20on%2028%20June%202023..

[292] Abdulrahman Abu Elkhail et al. "Vehicle security: A survey of security issues and vulnerabilities, malware attacks and defenses". In: *IEEE Access* 9 (2021), pp. 162401–162437.

[293] Simon Nam Thanh Vu et al. "A survey on botnets: Incentives, evolution, detection and current trends". In: *Future Internet* 13.8 (2021), p. 198.

[294] Harshit Gujral, Sangeeta Mittal, and Abhinav Sharma. "A novel data mining approach for analysis and pattern recognition of active fingerprinting components". In: *Wireless Personal Communications* 105 (2019), pp. 1039–1068.

[295] WAAA Bul'ajoul. "Performance of Network Intrusion Detection and Prevention Systems in Highspeed Environments". PhD thesis. Coventry University, 2017.

[296] Rongbo Zhang et al. "An Anonymous System Based on Random Virtual Proxy Mutation". In: *Tehnički vjesnik* 27.4 (2020), pp. 1115–1125.

[297] Sean Meek, Ivan Rodrigue Holguin, and Sanchari Das. "Can Johnny Really be Anonymous? Evaluation of User Data Privacy Within Tor". In: *Proceedings of the 6th Workshop on Technology and Consumer Protection (ConPro'22) Co-located with the 43th IEEE Symposium on Security and Privacy (IEEE S&P)*. 2022.

[298] Tor Metrics. "Tor Metrics - Servers". In: (2023). URL: https://metrics.torproject.org/networksize.html.%20Accessed%20on%2028%20June%202023.

[299] Tom Fawcett. "An introduction to ROC analysis". In: *Pattern recognition letters* 27.8 (2006), pp. 861–874.

[300] Geoffrey E Hinton and Sam Roweis. "Stochastic neighbor embedding". In: *Advances in neural information processing systems* 15 (2002).

[301] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[302] Longkang Shang et al. "Discovering unknown advanced persistent threat using shared features mined by neural networks". In: *Computer Networks* 189 (2021), p. 107937.

[303] CERT-EU. "Threat Landscape Report June 2021". In: (June 2021). Accessed on 2023-12-07. URL: https://media.cert.europa.eu/static/MEMO/2021/TLP-WHITE-CERT-EU-Threat_Landscape_Report-Volume1.pdf.

[304] Statista. "Revenue from advanced persistent threat (APT) protection market worldwide from 2015 to 2026". In: (Mar. 2022). Accessed on 2023-12-07. URL: https://www.statista.com/statistics/497945/advanced-persistent-threat-market-worldwide/.

[305] Gatewatcher. "The Rise of Advanced Persistent Threats". In: (May 2023). Accessed on 2023-07-23. URL: https://www.supplychainit.com/gatewatcher-unveils-research-into-advanced-persistent-threats/.

[306] Hamed Haddadpajouh et al. "MVFCC: A multi-view fuzzy consensus clustering model for malware threat attribution". In: *IEEE Access* 8 (2020), pp. 139188–139198.

[307] S Oliver. *MAC address randomization joins Apple's heap of iOS 8 privacy improvements*. 2014.

[308] Spyros Antonatos et al. "Defending against hitlist worms using network address space randomization". In: *Proceedings of the 2005 ACM workshop on Rapid malcode*. 2005, pp. 30–40.

[309] Dorene Kewley et al. "Dynamic approaches to thwart adversary intelligence gathering". In: *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*. Vol. 1. IEEE. 2001, pp. 176–185.

[310] Muhammad Salman Khan, Sana Siddiqui, and Ken Ferens. "A cognitive and concurrent cyber kill chain model". In: *Computer and Network Security Essentials* (2018), pp. 585–602.

[311] NIST. "NIST Special Publication 800-39: Managing Information Security Risk Organization, Mission, and Information System View". In: (2011). Accessed on 2023-07-3. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-39.pdf.

[312] CISA: Cybersecurity & Infrastructure Security Agency. "Cybersecurity Advisory: APT Groups Target Healthcare and Essential Services". In: (Jan. 2022). Accessed on 2023-12-10. URL: https://www.cisa.gov/news-events/cybersecurity-advisories/aa20-126a.

[313] Yong Fang et al. "LMTracker: Lateral movement path detection based on heterogeneous graph embedding". In: *Neurocomputing* 474 (2022), pp. 37–47.

[314] Mirco Marchetti et al. "Countering Advanced Persistent Threats through security intelligence and big data analytics". In: *2016 8th International Conference on Cyber Conflict (CyCon)*. IEEE. 2016, pp. 243–261.

[315] Akshaya Mani and Ian Goldberg. "ZXAD: High-volume Attack Mitigation for Tor". In: *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*. 2021, pp. 1–16.

[316] Igor Korkin and Iwan Nesterow. "Acceleration of statistical detection of zero-day malware in the memory dump using CUDA-enabled GPU hardware". In: *arXiv preprint arXiv:1606.04662* (2016).

[317] Sandro Skansi. *Introduction to Deep Learning: from logical calculus to artificial intelligence*. Springer, 2018.

[318] Martin T Hagan et al. "Training recurrent networks for filtering and control". In: *Recurrent neural networks: Design and applications*. CRC press, 1999, pp. 311–340.

[319] Gang Chen. "A gentle tutorial of recurrent neural network with error backpropagation". In: *arXiv preprint arXiv:1610.02583* (2016).

Appendix A

Author's Publications

***

A.1 Book Chapter

- Eke, Hope Nkiruka, Andrei Petrovski, Hatem Ahriz, and M. Omar Al-Kadri (2022). "Framework for Detecting APTs Based on Steps Analysis and Correlation." In Security and Resilience in Cyber-Physical Systems, pp. 119-147. Springer, Cham, 2022. Available from: https://doi.org/10.1007/978-3-030-97166-3_6

A.2 Conferences

- Eke, Hope Nkiruka., & Petrovski, Andrei. (2023, May). Advanced Persistent Threats Detection based on Deep Learning Approach. In 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS) (pp. 1-10). IEEE. Available from: https://doi.org/10.1109/ICPS58381.2023.10128062

- Eke, Hope, Andrei Petrovski, and Hatem Ahriz (2020a, November). "Detection of false command and response injection attacks for cyber physical systems security and resilience". In: 13th International Conference on Security of Information and Networks, pp 1-8. Available from: https://doi.org/10.1145/3433174.3433615

- Eke, Hope Nkiruka, Andrei Petrovski, and Hatem Ahriz (2019, September). "The use of machine learning algorithms for detecting advanced persistent threats". In: Proceedings of the 12th International Conference on Security of Information and Networks, pp. 1-8. Available from: https://doi.org/10.1145/3357613.3357618

A.3 Journals

- Eke, Hope, Andrei Petrovski, Hatem Ahriz (2020b). "Handling minority class problem in threats detection based on heterogeneous ensemble learning approach". In: International Journal of Systems and Software Security and Protection (IJSSSP) 11(2), pp. 13-37. Available from: https://doi.org/10.4018/IJSSSP.2020070102

A.4 Symposium

- Eke, Hope Nkiruka, Andrei Petrovski, and Hatem Ahriz (2019). "SYMPOSIUM: Securing Information Systems against Advanced Persistent Threats (APTs)". The Graduate School Symposium. Graduate School Symposium 19 June 2019.

A.5 Poster

- Eke, Hope Nkiruka, Andrei Petrovski, and Hatem Ahriz (2018). "POSTER: Securing Information Systems in Oil and Gas Industry against Advanced Persistent Threat (APT)". The Scottish Informatics & Computer Science Alliance (SICSA) PhD Conference 28-29 June 2018. The recorded video can be accessed from https://www.youtube.com/watch?v=tSdoHhcLoXo

Appendix B

Derivative of Los Function

***

This appendix contains the partial derivative[1] of Loss L function (see: Equation (2.10)) with respect to the network parameters.

To derive the derivative of loss function L in Equation (2.10) with respect to the actual true predicted class value o_t as calculated with Equation 2.9. Assume L as the objective function, $L_{(t)}$ as output at current time t and $L_{(t+1)}$ as the output at time $t+1$, such that $L_{(t+1)} = y_{(t+1)} log o_{(t+1)}$. The chain rule is utilized to perform the error backpropagation as to derive the minimum partial derivative of this loss function of the hidden state. The formula is expressed as follows;

$$\frac{\partial L}{\partial o_t} = -\sum_t y_t \frac{\partial log\ o_t}{\partial s_t} = -\sum_t y_t \frac{1}{o_t} \frac{\partial o_t}{\partial s_t} \tag{B.1}$$

From Equation (2.9), let's set $\theta = (W_{so} s_t + b_0)$, to have $o_t = sf(\theta)$. Applying the chain rule to derive the gradient of the sf in with respect to θ, and arrive at Equation (B.2).

$$\frac{\partial L}{\partial o_t} = -\left(y_t - o_t\right) \tag{B.2}$$

Considering that the activation of each unit in a softmax layer depends on the network input to every unit in the layer. Since the hidden state s and output o share the same weight W_{so} across the whole sequence in each time step t reducing the amount of parameter to be trained, weight can be differentiated at each time step, then summarize all together to achieve Equation (B.3).

$$\frac{\partial L}{\partial W_{so}} = \sum_t \frac{\partial L}{\partial o_t} \frac{\partial o_t}{\partial W_{so}} \tag{B.3}$$

Then, derive the gradient with respect to output bias b_o term to achieve Equation (B.4).

$$\frac{\partial L}{\partial b_o} = \sum_t \frac{\partial L}{\partial o_t} \frac{\partial o_t}{\partial b_o} \tag{B.4}$$

Also, let's consider the previous time step $t \rightarrow (t+1)$ in Figure (2.12) to derive the gradient with respect to weight W_{ss} as Equation (B.5).

$$\frac{\partial L(t+1)}{\partial W_{ss}} = \frac{\partial L(t+1)}{\partial o_{t+1}} \frac{\partial o_{t+1}}{\partial s_{t+1}} \frac{\partial s_{t+1}}{\partial W_{ss}} \tag{B.5}$$

To further the partial derivative with respect to W_{ss}, where Equation (B.5) only consider the time step $t \rightarrow (t+1)$. Also, considering that weight W_{ss} and W_{os} shared across all the time step t sequence are similar as indicated in Equations (2.8). Likewise, in RNN

[1] A detailed description of the derivation of these equations can be find in [199, 317, 318, 319] for a indepth understanding.

model, calculation of the subsequent hidden state s_t depends partially on the previous hidden state s_{t-1}, BBTT can be applied to compute partial derivative of Equation (B.5). Thus, at time step $(t-1) \rightarrow t$, to arrive at Equation (B.6).

$$\frac{\partial L(t+1)}{\partial W_{ss}} = \frac{\partial L(t+1)}{\partial o_{t+1}} \frac{\partial o_{t+1}}{\partial s_{t+1}} \frac{\partial s_{t+1}}{\partial s_t} \frac{\partial s_t}{\partial W_{ss}} \tag{B.6}$$

Hence, the gradient with respect to o_{t+1} at time step $t+1$ can be calculated, then BPTT is applied from t to 0 to calculate the gradient with respect to W_{ss} as illustrated with backward blue arrow in Figure (2.12). However, if output o_{t+1} only is considered at time step $t+1$, Equation (B.7) gradient can be achieved with respect to W_{ss} as;

$$\frac{\partial L(t+1)}{\partial W_{ss}} = \sum_{k=1}^{t} \frac{\partial L(t+1)}{\partial o_{t+1}} \frac{\partial o_{t+1}}{\partial s_{t+1}} \frac{\partial s_{t+1}}{\partial s_k} \frac{\partial s_k}{\partial W_{ss}} \tag{B.7}$$

Furthermore, BPTT was used to obtain the gradient as Equation (B.8) by clustering the gradient with respect to weight W_{ss}, over the whole time sequence as;

$$\frac{\partial L}{\partial W_{ss}} = \sum_{t} \sum_{k=1}^{t+1} \frac{\partial L(t+1)}{\partial o_{t+1}} \frac{\partial o_{t+1}}{\partial s_{t+1}} \frac{\partial s_{t+1}}{\partial s_k} \frac{\partial s_k}{\partial W_{ss}} \tag{B.8}$$

Adding up all the contributions from t to 0 using BBTT, gradient can be computed at time step $t+1$ and obtain (B.9).

$$\frac{\partial L(t+1)}{\partial W_{xs}} = \sum_{k=1}^{t+1} \frac{\partial L(t+1)}{\partial s_{t+1}} \frac{\partial s_{t+1}}{\partial s_k} \frac{\partial s_k}{\partial W_{xs}} \tag{B.9}$$

Taking the gradient with respect to W_{xs} over the whole sequence, applied the same process as from Equation (B.2) to (B.9) and get Equation (B.10).

$$\frac{\partial L}{\partial W_{xs}} = \sum_{t} \sum_{k=1}^{t+1} \frac{\partial L(t+1)}{\partial o_{t+1}} \frac{\partial o_{t+1}}{\partial s_{t+1}} \frac{\partial s_{t+1}}{\partial s_k} \frac{\partial s_k}{\partial W_{xs}} \tag{B.10}$$

In Equation (B.10), $\frac{\partial s_{t+1}}{\partial s_k}$ indicates matrix multiplication over the sequence. RNN suffer from vanishing and exploding gradient problems. Gradient value become smaller and will eventually vanish after a few time steps as RNN backpropagate gradients over a long sequence. Thus, the farther states from the current time step does not make any contribution towards computing the parameters gradient [319].

Appendix C

Codes and Snippets

***

C.1 Code Snippets

```
1
2  # Pre-processing Imports
3  from __future__ import print_function
4  from sklearn import preprocessing
5  import matplotlib.pyplot as plt
6  import numpy as np
7  import pandas as pd
8  import requests
9  import pickle
10 import shutil
11 import base64
12 import os
13 import csv
14 from collections import OrderedDict
15
16 # Process data
17 from process_data import process_data as pro
18
19 # Cnfusion Matrix
20 from sklearn.metrics import confusion_matrix
21
22 # ROC Curve
23 from sklearn.preprocessing import label_binarize
24 from scipy import interp
25 from itertools import cycle
26
27 # Decision Tree
28 from sklearn.tree import DecisionTreeClassifier
29
30 # Model Imports
31 from time import time
32 from sklearn.utils import class_weight
33 from sklearn.model_selection import train_test_split
34 import tensorflow as tf
35 import sys
36 np.random.seed(1337)  # for reproducibility
37 from keras.preprocessing import sequence
38 from keras.models import load_model
39 from keras.utils import np_utils
40 from keras.models import Sequential
41 from keras.layers import Dense, Dropout, Activation, Embedding
42 from keras.datasets import imdb
43 from keras.utils.np_utils import to_categorical
44 from sklearn.metrics import (precision_score, recall_score,
```

```
                                f1_score, accuracy_score,
    mean_squared_error,mean_absolute_error)
from sklearn import metrics
from scipy.stats import zscore
from sklearn import preprocessing
from sklearn.preprocessing import Normalizer
import h5py
from keras import callbacks
from keras.callbacks import ModelCheckpoint, EarlyStopping,
    ReduceLROnPlateau, CSVLogger
```

LISTING C.1: Used Libraries

```

# Encode text values to dummy variables (i.e. [1,0,0], [0,1,0],[0,0,1]
    for proto, service, state) using get_dummies() function.

def encode_text_dummy(df, name):
    dummies = pd.get_dummies(df[name])

    for x in dummies.columns:
        dummy_name = "{}-{}".format(name, x)
        df[dummy_name] = dummies[x]

    df.drop(name, axis=1, inplace=True)
```

LISTING C.2: Feature Transformation

```

# Balance data in both training and testing set to have the same number
     of columns using "Union Set"

def balance_df(train_data, test_data):
    train_data_columns = list(train_data)
    test_data_columns = list(test_data)
    column_union = list(set().union(train_data, test_data))

    for i in column_union:
        if i not in list(train_data):
            train_data[i] = 0

        if i not in list(test_data):
            test_data[i] = 0
```

LISTING C.3: Balancing Data Features

```

# Encode a numeric column as zscores - This was used to normalize all
    the numeric data

def encode_numeric_zscore(df, name, mean=None, sd=None):
    if mean is None:
        mean = df[name].mean()

    if sd is None:
        sd = df[name].std()

    df[name] = (df[name] - mean) / sd

    return df, mean, sd
```

LISTING C.4: Data Normalization

```
1
2 # LabelEncoder(): Encode text values to indexes (i.e. [1], [2], [3] for
      Normal, Backdoor and DoS)
3
4 def encode_text_index(df, name):
5
6     le = preprocessing.LabelEncoder()
7     df[name] = le.fit_transform(df[name])
8
9     return le.classes
```

LISTING C.5: Label Encoder

```
1
2 # Confusion Matrix Function
3 from sklearn.metrics import confusion_matrix
4 def plot_confusion_matrix(y_true, y_pred, classes, algo, cmap=plt.cm.
    Purples):
5
6     print(y_true)
7     print(y_pred)
8     # Compute confusion matrix
9     cm = confusion_matrix(y_true, y_pred)
10    print(cm)
11    fig, ax = plt.subplots(figsize=[8,8])
12    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
13    ax.figure.colorbar(im, ax=ax)
14
15    ax.set(xticks=np.arange(cm.shape[1]),
16           yticks=np.arange(cm.shape[0]),
17           # label with the respective list entries
18           xticklabels=classes, yticklabels=classes,
19           title="Confusion Matrix for " + algo,
20           ylabel='True label',
21           xlabel='Predicted label')
22
23    # Rotate the tick labels and set their alignment.
24    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
25             rotation_mode="anchor")
26    plt.rcParams.update({'font.size':10})
27
28    # Loop over data dimensions and create text annotations.
29    fmt = 'd'
30    thresh = cm.max() / 2.
31    for i in range(cm.shape[0]):
32        for j in range(cm.shape[1]):
33            ax.text(j, i, format(cm[i, j], fmt),
34                    ha="center", va="center",fontweight='bold',
35                    color="white" if cm[i, j] > thresh else "black")
36    fig.tight_layout()
37    plt.savefig("./ngpresults/dl_cat/cm_images/" + algo +"
    _ngp_cat_cm_epochs.png",dpi=300)
38    return cm
```

LISTING C.6: Confusion Matrix Function

```
# Pre-processing Module

def process_data(df, missing_val_id='?'):
    # Replace missing values
    dfc = df.replace('?', 1)

    for i in list(dfc):
        if i != 'categorized_result':
            # print(i)
            dfc[i] = dfc[i].astype(float)

    cols = list(dfc)
    cols.remove('categorized_result')

    stats = pd.read_csv('./data/4b_ngp_cat-stats.csv')

    for i in cols:
        # print(i)
        m = stats[i][0]
        s = stats[i][1]
        dfc, _, _ = encode_numeric_zscore(dfc, i, mean=m, sd=s)

    x = np.array(dfc.iloc[:, dfc.columns != 'categorized_result'])
    y = np.array(dfc['categorized_result'])
    scaler = pickle.load(open('./data/4b_ngp_cat-trained_scaler.pkl',
                              'rb'))
    X = scaler.transform(x)

    return X, y
```

LISTING C.7: Process_Data Module

```
# Function to enumeration instances within the dataset

import matplotlib.pyplot as plt
import numpy as np

labels = cat_labels
no_instances = cat

#def plot_instances_graphs(history, algo):
index = np.arange(len(labels))
print(index) # [0 1 2 3 4 5 6 7]
plt.figure(figsize=[9,5])
plt.barh(index, no_instances)#, align='center', alpha=0.8)
plt.ylabel('Classes')
plt.xlabel('Number of Class Instances')
plt.yticks(index, labels)
plt.title('7 Categorized Attack Type Group and Normal')
plt.rcParams.update({'font.size':10})
#plt.tight_layout()
fig = plt.gcf()
for i, v in enumerate(no_instances):
    plt.text(v, i, " "+str(v), color='purple', va='center', fontweight=
    'bold') ##D63B59
    plt.savefig("./enumarate_ngp/ngp_categorized_label.png")
plt.show()
```

LISTING C.8: Enumeration_Function Module

```
1
2 # ROC function for single-class
3
4 def plot_roc_curve_single_class(testY, y_pred, class_num, algo):
5     from sklearn.metrics import roc_curve, auc
6     fpr = dict()
7     tpr = dict()
8     roc_auc = dict()
9     for i in range(len(testY[0])):
10         fpr[i], tpr[i], _ = roc_curve(testY[:, i], y_pred1[:, i])
11         roc_auc[i] = auc(fpr[i], tpr[i])
12
13     # Compute micro-average ROC curve and ROC area
14     fpr["micro"], tpr["micro"], _ = roc_curve(testY.ravel(), y_pred1.
    ravel())
15     roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
16
17     plt.figure()
18     lw = 2
19     plt.plot(fpr[class_num], tpr[class_num], color='navy',
20              lw=lw, label='ROC curve (area = %0.2f)' % roc_auc[
    class_num])
21     # plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
22     plt.xlim([0.0, 1.0])
23     plt.ylim([0.0, 1.05])
24     plt.xlabel('False Positive Rate')
25     plt.ylabel('True Positive Rate')
26     plt.title('Receiver Operating Characteristic example for class ' +
    str(class_num))
27     plt.legend(loc="lower right")
28     plt.savefig("./ngpresults/dl_cat/roc_auc/" + algo + "
    _roc_curve_for_classNo_" + str(class_num) + ".png",dpi=300)
29     plt.show()
```

LISTING C.9: ROC for Single-Class Module

```
1
2 # ROC function for multi-class
3
4 def plot_roc_curve(testY, y_pred, algo):
5     from sklearn.metrics import roc_curve, auc
6     fpr = dict()
7     tpr = dict()
8     roc_auc = dict()
9     for i in range(len(testY[0])):
10         fpr[i], tpr[i], _ = roc_curve(testY[:, i], y_pred1[:, i])
11         roc_auc[i] = auc(fpr[i], tpr[i])
12
13     # Compute micro-average ROC curve and ROC area
14     fpr["micro"], tpr["micro"], _ = roc_curve(testY.ravel(), y_pred1.
    ravel())
15     roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
16
17     n_classes = len(testY[1])
18     # Compute macro-average ROC curve and ROC area
19
20     # First aggregate all false positive rates
21     all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes
    )]))
22
23     # Then interpolate all ROC curves at this points
24     mean_tpr = np.zeros_like(all_fpr)
25     for i in range(n_classes):
```

```
        mean_tpr += interp(all_fpr, fpr[i], tpr[i])

    # Finally average it and compute AUC
    mean_tpr /= n_classes

    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    # Plot all ROC curves
    plt.figure(figsize = [15, 10])
    lw = 2
    plt.plot(fpr["micro"], tpr["micro"],
             label='micro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["micro"]),
             color='deeppink', linestyle=':', linewidth=4)

    plt.plot(fpr["macro"], tpr["macro"],
             label='macro-average ROC curve (area = {0:0.2f})'
                   ''.format(roc_auc["macro"]),
             color='navy', linestyle=':', linewidth=4)

    colors = cycle(['indigo', 'yellow', 'green', 'purple', 'orange', '
    darkblue', 'aqua', 'darkorange'])
    for i, color in zip(range(n_classes), colors):
        plt.plot(fpr[i], tpr[i], color=color, lw=lw,
                 label='ROC curve of class {0} (area = {1:0.2f})'
                 ''.format(i, roc_auc[i]))

    plt.plot([0, 1], [0, 1], 'k--', lw=lw)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) for 8 Classes')
    plt.legend(loc="lower right")
    plt.savefig('./ngpresults/dl_cat/roc_auc/' + algo + '
    _ngpcat_roc_curve_epochs.png',dpi=300)
    plt.show()
```

LISTING C.10: ROC for Multi-Class Module

```
1
2 # Plot the graphs for loss and accuracy
3
4 import matplotlib.pyplot as plt
5
6 def plot_loss_graphs(history, algo):
7     loss = history.history['loss']
8     val_loss = history.history['val_loss']
9     epochs = range(1, len(loss) + 1)
10    plt.plot(epochs, loss, label='Training loss')
11    plt.plot(epochs, val_loss, label='Validation loss')
12    plt.title(algo + ' - Training and validation loss')
13    plt.xlabel('Epochs')
14    plt.ylabel('Loss')
15    plt.legend()
16    plt.savefig("./ngpresults/dl_cat/acc_loss/" + algo+"
      _ngpcat_loss_epochs.png",dpi=300)
17    plt.show()
18
19 def plot_acc_graphs(history, algo): # algo is used to pass the name to
      function
20    acc = history.history['accuracy']
21    val_acc = history.history['val_accuracy']
22    epochs = range(1, len(acc) + 1)
23    plt.plot(epochs, acc, label='Training acc')
24    plt.plot(epochs, val_acc, label='Validation acc')
25    plt.title(algo + ' - Training and validation accuracy')
26    plt.xlabel('Epochs')
27    plt.ylabel('Accuracy')
28    plt.legend()
29    plt.savefig("./ngpresults/dl_cat/acc_loss/" + algo+"
      _ngpcat_acc_epochs.png",dpi=300)
30    plt.show()
```

LISTING C.11: Function to Plot Graphs for Loss and Accuracy

```
1
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 N = 8 # no of classes in the datasets
6
7 ind = np.arange(N)  # the x locations for the groups
8 width = 0.3       # the width of the bars
9
10 fig = plt.figure(figsize=[45, 20])
11 plt.rcParams.update({"font.size":30})
12 plt.ylim([0, 1.2])
13 ax = fig.add_subplot(111)
14
15 lstmvals = lstm # lstm value ie predicted accuracy
16 rects1 = ax.bar(ind, lstmvals, width, color='navy')
17 gruvals = gru # gru values
18 rects2 = ax.bar(ind+width, gruvals, width, color='g')
19 rnnvals = rnn # rnn values
20 rects3 = ax.bar(ind+width*2, rnnvals, width, color='y')
21 #cnnvals = cnn
22 #rects4 = ax.bar(ind+width*3, cnnvals, width, color='aqua')
23
24 ax.set_ylabel('Accuracy')
25 ax.set_xlabel('Classes')
26 plt.title('Perfomance Accuracy of Each Algorithm on all Five Classes  '
      )
```

```
ax.set_xticks(ind+width)
ax.set_xticklabels(class_names)
ax.legend( (rects1[0], rects2[0], rects3[0]), ('LSTM', 'GRU', 'RNN'))
#ax.legend( (rects1[0], rects2[0], rects3[0], rects4[0]), ('LSTM', 'GRU', 'RNN', 'CNN'))

def autolabel(rects):
    for rect in rects:
        h = rect.get_height()
        ax.text(rect.get_x()+rect.get_width()/2., 1.05*h, '%.3f'%float(h),
                ha='center', va='bottom', fontweight='bold')

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
fig.tight_layout()
plt.savefig("./ngpresultsdtb/com_res_inject/cm_images/ngp_acc_classes5-3.tiff",dpi=300)
plt.show()
```

LISTING C.12: Prediction Graph Function for Multi-classes Module

```

#  Ensemble by Voting function

def voting_classif(X, y, est): # binary calssification, highest no of votes
    pred_y = {}
    for j in est:
        #print("\n this is classifier", j[0])
        predict = j[1].predict_classes(X)
        pred_y[j[0]] = predict

    print("Predictions Done!")
    print("==================")
    pred_df = pd.DataFrame(pred_y, columns = est[:, 0])
    predictions = np.array(pred_df.mode(axis='columns')[0])
    #print("mode done")
    acc = metrics.accuracy_score(predictions, y)
    print("accuracy is", acc)
    m_f1 = metrics.f1_score(predictions, y, average="macro")
    print("macro f1 is", m_f1)
    return acc,

# Ensemble prbability function

def prob_classif(X, y, est):  # multi-clasiffication
    pred_y = []
    for i in est:
        pred_i = i[1].predict_proba(X)
        pred_y.append(pred_i)
    p = np.array(pred_y)
    meta_data = np.swapaxes(p, 0, 1)
    sum_prob = sum(p)
    preds = sum_prob.argmax(axis=1)
    predictions = np.array(preds)
    acc = metrics.accuracy_score(predictions, y)
    return acc, predictions
```

LISTING C.13: Ensemble Module

Index

***

Adam, 96
APT$_{DASAC}$, 6, 151
APTs, 1
Architectural Design, 82

CMRI, 62
Command injection attacks, 63
Computer aided diagnosis, 38
cross-entropy loss, 96

Data analysis layer, 83
Data input and probing layer, 83
Decision Layer, 83
DoS attacks, 64
dropout, 97

Fabrication, 64

Interception, 64
Interruption, 64

MFCI, 63
Modification, 64
MPCI, 63
MSCI, 63
Myocardial infarction, 38

network topologies, 62, 105
NGP dataset, 83
NGP Raw Dataset, 64
NMRI, 62

parameterizing, 64
payload information, 62, 105
pseudocode, 83

randomizing, 64
Response injection attacks, 62

sigmoid/ReLU, 96
softmax, 96

UNSW-NB15 dataset, 83