Towards situational awareness of botnet activity in the Internet of Things.

MCDERMOTT, C.D., PETROVSKI, A.V. and MAJDANI, F.

2018

Towards Situational Awareness of Botnet Activity in the Internet of Things

Christopher D. McDermott, Andrei V. Petrovski, Farzan Majdani

School of Computing Science and Digital Media
Robert Gordon University
Aberdeen, United Kingdom
Emails: {c.d.mcdermott, a.petrovski, f.majdani-shabestari}@rgu.ac.uk

***Abstract*—An IoT botnet detection model is designed to detect anomalous attack traffic utilised by the mirai botnet malware. The model uses a novel application of Deep Bidirectional Long Short Term Memory based Recurrent Neural Network (BLSTM-RNN), in conjunction with Word Embedding, to convert string data found in captured packets, into a format usable by the BLSTM-RNN. In doing so, this paper presents a solution to the problem of detecting and making consumers situationally aware when their IoT devices are infected, and forms part of a botnet. The proposed model addresses the issue of detection, and returns high accuracy and low loss metrics for four attack vectors used by the *mirai* botnet malware, with only one attack vector shown to be difficult to detect and predict. A labelled dataset was generated and used for all experiments, to test and validate the accuracy and data loss in the detection model. This dataset is available upon request.**

***Index Terms*—Situational Awareness, Long Short Term Memory Networks, Deep Learning, IoT, Botnet, Mirai, DDoS, Word Embedding.**

I. Introduction

The Internet of Things (IoT) has quickly transitioned from a promising future paradigm to a pervasive everyday reality. Billions of smart devices are now being connected to the Internet creating an extensive network of connected *'things'*, capable of sensing the surrounding environment and interacting with other devices, to aid real-time monitoring and decision making [1]. The potential benefits of the IoT are clearly evident, however despite the promise of new business models, increased efficiency, and the enrichment of everyday life, concerns still remain regarding security and privacy. Many of these concerns arise as a result of device manufacturers excluding security and privacy mechanisms from their products, following market pressure to produce cheap plug and play connected products [2]. The result has been a rapid proliferation of insecure connected devices across the internet, and the danger of another paradigm shift to an *Internet of Insecure Things* [3].

One of the biggest threats derived from the rise in IoT devices, is that of botnets. The huge number of connected devices, coupled with their inherent security issues, has provided the perfect tool to carry out powerful distributed denial of service (DDoS) attacks [3]. These attacks often target consumer level products, which commonly lack a screen or user interface, making it difficult to identify signs of infection or detect attacks which occur or emanate from within the home network. To evaluate this problem a sandboxed botnet environment was built as described in Section III, and a simple series of experiments performed. An IoT IP camera was infected and became part of a botnet. The camera was instructed to perform various DDoS Attacks, during which live video feeds were recorded, to ascertain if any signs of infection could be detected. The recorded feed for each DDoS attack was presented to a small test group, along with video feeds of uninfected IP cameras, and the participants were asked to identify any unusual activity within the video feeds. In all experiments the participants were not able to identify the video feeds recorded on infected cameras, or when an attack was taking place. Without any clear signs of infection, these experiments demonstrate that it is very difficult for consumers to know when their devices are part of a botnet, performing large scale DDoS attacks.

Motivated by this problem, and to improve situational awareness of such attacks, this paper presents a solution to the detection of botnet activity within consumer IoT devices and networks. Computational intelligence techniques have been tested for their effectiveness in identifying anomalies in network traffic. We implemented our approach by using the experimental botnet framework, infecting an IoT IP Camera, and using the camera to perform a series of DDoS attacks against a victim device. A labelled dataset was created and used to test a detection model based on a Deep Bidirectional Long Short Term Memory based Recurrent Neural Network (BLSTM-RNN), in conjunction with Word Embedding, to convert string data into a format usable by the BLSTM-RNN. Thus, the main contributions of this paper are:

1) A labelled dataset encompassing botnet activity and DDoS attacks;
2) A detection algorithm which utilises a word embedding methodology to extract intelligence from textual features, and parse the data into a format useable by a BLSTM-RNN;
3) A detection model to detect and predict infected IoT device traffic.

The rest of the paper is organized as follows: Section II introduces related work and the issue of situational awareness of botnet activity within the IoT. Section III details our approach to IoT botnet detection using a combination of BLSTM-RNN in conjuction with Word Embedding methodology. It also describes the experimental set-up used to generate the botnet dataset. Section IV presents the experiments undertaken and provides discussion on the results gained. Section V draws final conclusions and speculates possible future work.

II. Background

For our related works, we shall consider the topics of botnets in the IoT, botnet detection methods, and situational awareness of botnet activity by non-expert users (NEU). Although botnet activity and detection has been well researched, the existing literature generally focuses on traditional network botnets, rather than specifically focusing on botnets that target the IoT. Conference papers and peer reviewed articles presented below therefore either directly target IoT botnets, or could easily be applied to this research area.

A. Botnets in the Internet of Things

Some of the most extensive and destructive cyber-attacks deployed on the Internet have been Distributed Denial of Service (DDoS) attacks. According to Akamai, a global leader in web security, some of the largest DDoS attacks ever recorded occurred in the second half of 2016, fuelled in full or part by the Internet of Things [4]. During this time, attacks of over 100Gbps were up by 140%, with three attacks reaching over 300 Gbps. The severity of the attacks continued in 2017, evidenced in Versign's annual DDoS Trends report which reported that 88% of recorded DDoS attacks in quarter 4 of 2017 also now employed a multi-vector attack strategy [5].

In [6] the authors present several IoT botnets, which exploit basic security vulnerabilities, and show how these types of botnets are becoming increasingly more sophisticated in their effectiveness and ability to obfuscate their activity. They present *MalwareMustDie* as an example which uses iptables rules to protect its infected devices, whilst *Hajime* utilises fully distributed communications and makes use of the BitTorrent protocol for peer discovery. *BrickerBot* was also presented which leverages SSH default credentials to perform a permanent denial-of-service (PDoS) attack.

In [7] [8] one of the most prominent examples of a DDoS attack emanating from the IoT in recent times, is presented. Mirai is a piece of malware that attempts to find and infect IoT devices to establish and propagate a network of robots (botnet) consisting of the infected IoT devices (bots). An attacker (botmaster) then uses a command and control (C&C) server to remotely control the bots, forcing them to participate in DDoS attacks against targets on the Internet. On September 20 2016 the Mirai botnet was used to perform an unprecedented 620 Gbps DDoS attack on security journalist Brian Krebs website krebsonsecurity.com [9]. Shortly after it was also responsible for a series of additional DDoS attacks peaking at over 1.2 Tbps against French hosting company OVH and DNS provider DYN, who estimated that up to 100 000 infected IoT devices (bots) were involved in the attack. The severity of the DYN attack was sufficient to cause major disruption on the Internet, and render several high profile websites such as GitHub, Twitter, Reddit, Netflix, inaccessible.

To foster greater situational awareness of how botnets target insecure IoT devices, we conducted a detailed analysis of the Mirai botnet, which will be presented below. Fig. 1 shows the process of infection and propagation method employed by Mirai. The Mirai infrastructure consists of a command and control (C&C) server, a Scan/Loader server and infected IoT devices known as bots.

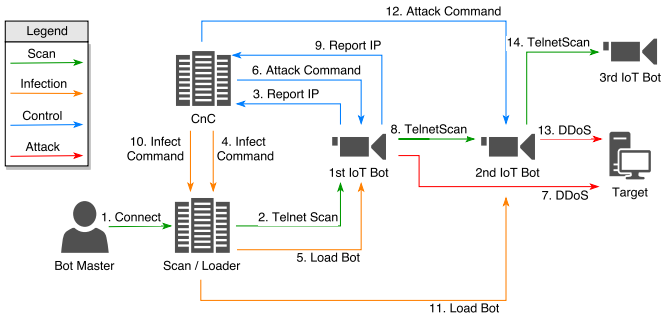

Fig. 1. Mirai Botnet Infection Methodology

Infection and propagation occurs by exploiting weak default security credentials found on many IoT devices running *busybox*, an embedded version of Linux. An attacker (botmaster) starts the process by connecting to the Scan/Loader server (*step 1*) and initiating *./loader* to execute the *scanner.c* module, and scan the Internet for vulnerable IoT devices with Telnet services and ports 23 or 2323 open (*step 2*). Upon detecting a vulnerable device, the malware attempts to brute force a successful login using a list of 62 known default usernames and passwords. If successful, login credentials and device information are sent back to the C&C server, and will be used later by the Scan/Loader server to login and deliver the malware to the vulnerable device (*step 3*). An infect command is sent from the C&C server to the Scan/Loader server containing all necessary information such as login details, IP address, hardware architecture. Mirai supports multiple hardware architectures, including *arm*, *mips*, *sparc* and *powerpc* (*step 4*).

The Scan/Loader server uses this information to login and instruct the vulnerable device to *tftp* or *wget* to the Scan/Loader server, download and execute the corresponding payload binary. Once executed, the first infected IoT device becomes part of the Mirai botnet and can communicate with the C&C server. The malware binary is removed and runs only in memory, to avoid detection (*step 5*). The botmaster can now issue attack commands, specifying parameters such as attack duration and target (*step 6*). The malware includes 10 DDoS attack types, including UDP flood (*udp*), Recursive DNS (*dns*), SYN packet flood (*syn*), ACK packet flood (*ack*), GRE flood (*gre ip*), which can be used to attack a target on the Internet (*step 7*). The first bot now attempts to repeat

the infection process and propagate the botnet by scanning the Internet for additional vulnerable IoT devices with Telnet services and ports 23 or 2323 open (*step 8*). New vulnerable IoT device information is returned to the C&C server (*step 9*). A new infect command is issued to the Scan/Loader server (*step 10*). The appropriate hardware binary is loaded onto the newly discover vulnerable IoT device (s*tep 11*). The relevant attack command is issued from the C&C server (*step 12*). The attack is executed by the newly infected second bot, in conjunction with the first bot (*step 13*). Scanning for additional vulnerable IoT devices is repeated to further expand the botnet. (*step 14*).

B. Botnet Detection Methods

As previously stated much of the existing literature on botnet detection generally focuses on traditional network botnets, rather than IoT botnets. An increasingly popular approach has been the use of Machine Learning Algorithms (MLA) for network traffic analysis and classification. The assumption being that botnets create distinguishable traffic patterns, that can be used to accurately detect botnet activity [10].

In many cases traffic analysis was performed at the network level, analysing *flows* of traffic conversations, rather than at the individual packet level. In doing so, the authors in [11] used a Support Vector Machine, C4.5 decision tree and Random forest classifier to classify malicious and non-malicious in a NetFlow dataset, and harness true positive detection rates above 70%.

In [12] a decision tree using the Reduced Error Pruning Algorithm (REPTree) was used and again demonstrated good performance with true positive detection rates above 90%. It should be noted however, that the use of IP addresses as a prominent feature could result in an unbalanced dataset, and effect detection results.

In [13] the authors consider smaller packet correlation as a way of improving detection accuracy, by extracting additional features, namely *packet ratio*, *initial packet length*, and *bot response time*, and modelling the behaviour of network flows. Flows were classified using Boosted decision tree (AdaBoostM1+J48), Naive Bayesian (NB), and Support Vector Machine (SVM) algorithms, and the detection system tested against three separate datasets. The authors suggest the advantage of the proposed system is its lightweight nature, however this was not substantiated through comparison with alternative detection methods.

In [10] the authors compare eight MLAs for classifying botnet traffic, and also compare two scenarios for traffic analysis. In the first scenario flows are monitored in their entirety from start to end, whereas in the second scenario, traffic flows are only observed for a specific time interval and maximum number of packets. They successfully demonstrated that detection rates could be maintained whilst reducing sample sizes to only 10 packets and 60 seconds of monitored flow traffic. In [14] they extend their work and propose three methods of traffic analysis for botnet detection, utilising a Random Forest classifier on 40 different bot samples, classifying TCP, UDP and DNS communications separately. Classification accuracy for all protocols was above 90%, although balanced classification required a time of window length of 3600 seconds and 1000 packets, which could result in a lower detection rate for attacks with smaller sample metrics.

In [15] the authors propose *IoTSec*, which utilises a Software Defined Networking (SDN) approach to enforcing security policies for IoT devices. Whilst the proposal of a *crowd-sourced* repository of learned attack signatures, could be useful in detecting botnet actvity, it relies on NEUs providing this information, which would prove challenging.

The main drawback for many of these approaches is that they analyse traffic flows rather than individual packets, which results in only representative samples of the total traffic, being considered. In addition with regard to the problem highlighted in Section I, it is unlikely that consumer routers would have the ability to capture traffic flows using protocols such as *NetFlow* or *sFlow*, therefore many of the approaches may not be transferable to IoT botnet detection.

C. Situational Awareness of IoT botnet activity by Non-Expert Users

Situational awareness (SA) is often defined as the perception of environmental elements, comprehension of their meaning in context, and the projection of likely future events [16]. Applied in a cyber context it relates to the compilation, processing and fusing of network data to understand a network environment and accurately predict and respond to potential threats that might occur.

In [17] the authors suggest cyber SA is often recognised as important, but the ability to systematically evaluate and work on it is often limited. They propose a taxonomy to aid decision makers in structuring and reasoning about cyber security awareness in their context. Three essential elements are presented as necessary to achieve cyber situational awareness. *Data gathering*, from firewalls, anti-virus or vulnerability scanners; *Analysis*, through anomaly detection, parsing logs, or metrics; and *Visualisation*, consisting of statistical, historical and real-time presentation of data.

In [18] the author presents the need for greater online awareness and protection for NEUs. The author suggests a lack of technical knowledge and ability to explore network communication, results in little or no awareness of security issues. In response to this a security visualisation framework is proposed to support NEUs to engage with network traffic analysis in order to better support their perception and comprehension of cyber security concerns.

Despite research in this area it is clear from our preliminary research and information presented in Section II-A, that it is still difficult for NEUs to be situationally aware of their network environment and accurately detect and respond to threats posed by IoT botnets.

III. Methodology

To promote reproducibility of this paper, a detailed description of botnet environment and detection algorithm implementation is presented. A secure sandboxed environment was

created as shown in Fig. 2. This consisted of a command and control C&C server, a Scan/Loader server and an additional utilities server to handle DNS queries and reporting. A soft tap (Tap0) SPAN port was created to mirror all relevant traffic to a packet sniffing device, to capture for later analysis. Two Sricam AP009 IP Cameras running *busybox* utilities were used as bots to attack a target Raspberry Pi.

The Mirai source code was downloaded from GitHub. To ensure a true representation of a Mirai infection and attack, amendments to the source code were kept to a minimum however, some configuration changes were required to comply with ethical and legal regulations.

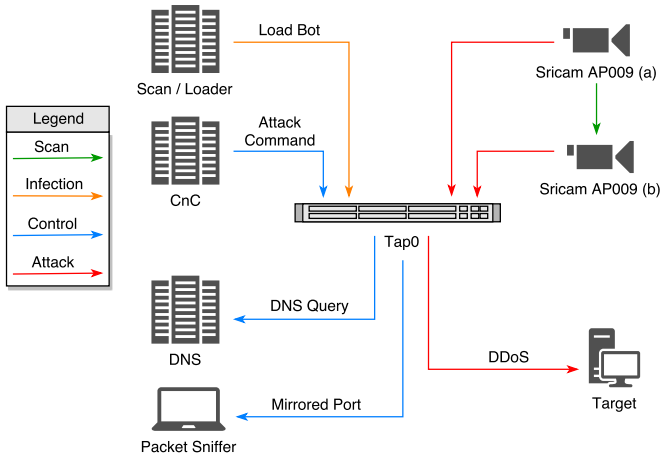

Fig. 2. Mirai Botnet Experimental Setup

1) C&C Server Configuration: Essential packages were installed using *apt-get install unzip gcc golang electric-fence screen y*

Domains were created for report.McDPhD.org and cnc.McDPhD.org, and added to *table.c* and *main.go*.

MySQL was installed using *apt-get install mysql-server mysql-client y* and a user created using *INSERT INTO users VALUES (NULL, 'miraiuser', 'miraipassword', 0, 0, 0, 0, -1, 1, 30, '')*; Once configured *main.go* was edited to include the MySQL credentials.

Cross compilers for the required binary architectures (e.g. *arm*, *mips*) were installed and appropriate export paths added to */etc/profile* using *export PATH= $PATH: /etc/xcompile/mips/bin*. To allow information regarding C&C connections, compiler issues and flood status to be sent the C&C server *./build.sh debug telnet* was run. The required binary files for each architecture were created and stored in the *release* directory using *./build.sh release*

2) Scan Loader Server Configuration: Apache was installed using *apt-get install apache2 y* and binary architecture files created earlier, were moved to the *loader/bins* directory. The Scan/Loader IP address was added to *main.c* and full permission granted using *chmod777**. The *loader* file was compiled and added to the loader directory using *./build.sh*

To reduce the number of IP ranges available for scanning and ensure the range used in our environment was allowed, excluded IP ranges were amended in *scanner.c* to reflect our topology.

The Scan/Loader IP address was added to *scanListen.go* and port *48101* specified as the default port to listen for brute force results. Within the *tools* directory the *scanListen* file was compiled using *go build scanListen.go* and moved to the *loader* directory.

The Sricam AP009 IP camera used in the lab setup did not include *wget*, therefore tftp was installed using *apt-get install tftpd tftp*. A tftp configuration was created using *touch /etc/xinetd.d/tftp* and */tftpboot* specified as the directory where the architecture binary files will be copied to for later delivering the payload.

3) DNS Server Configuration: The Mirai malware requires access to a DNS server to discover the C&C servers IP address. *Bind9* software was installed and used to create two required domains *report.McDPhD.org* and *cnc.McDPhD.org* in *named.conf.local*. These will be used by the bots to report IoT device information and communicate with the C&C server.

Our model focuses on analysing textual data within packets, overlooked by other detection methods, to detect infected IoT devices, and raise situational awareness of these types of threats. As a modular system design, our model is made up of two core modules:

A. Data Parser Module

The dataset used in our experiments was generated from the experimental set-up described in Section III. It consists of Mirai botnet traffic such as *Scan*, *Infect*, *Control* and *Attack* traffic as described in Section II-A and normal IoT IP Camera traffic generated in our experimental set-up. The dataset included features *No.*, *Time*, *Source*, *Destination*, *Protocol*, *Length*, and overall payload information in the *Info* feature. Some features such as *No.* and *Time* did not provide much scope for data analysis so were removed.

Majority of the captured information resided in the *Info* feature, as shown in Table I therefore a model was required which could read and understand the text presented in this feature. As discussed in Section III-B an Artificial Neural Network(ANN) and more complex versions of Recurrent Neural Networks(RNN) such as Long Short Term Memory (LSTM) only work with numerical values. However [19] demonstrated that a Deep Bidirectional Long Short Term Memory based RNN (BLSTM-RNN) can be used which provides promising results for text recognition. This potential was further demonstrated in [20] where a BLSTM-RNN was used in conjunction with Word Embedding, in such a way phrases and vocabulary were mapped to vectors or real numbers, and proved to be an effective method for modelling and predicting sequential text.

Motivated by this potential, this paper presents a BLSTM-RNN algorithm and model which is applied to botnet detection in the IoT. Since the information provided in the *Info* feature of the dataset follows a sequence, we implemented our approach by first converting each letter into a tokenized and integer encoded format.

A dictionary of all tokenized words and their index within the *Info* feature was created and text replaced with its corresponding index number. In order to understand each attack type, it was important to maintain the sequence order of the indices, therefore an array of the indices was created. Since attacks are often closely coupled to the protocol used and the length of the captured packet, the *Protocol* and *Length* features also required to be included in the array. Word Embedding was again used to convert and create a dictionary of all tokenized protocols and their index. These were then added, along with the *Length* feature, which was already an integer, to the array. Labels identifying each type of captured packets were mapped from string to integer ('norm': 0,'mirai':1,'udp': 2, 'dns':3, 'ack':4,'normal':5), and also injected into the array.

Algorithm 1 IoTBot Detection and Prediction

```
dataProcessing (dataset)
unitToDrop ← 25%
Parse data to predefined format
Define token dictionary
repeat
    /*Parse data to format*/
    for i ← 1, rows do
        Convert text to number using Word Embedding
        Add converted tokenised words into dictionary
        Pad data arrays with 0s
        Convert
        Inject other tokenised hyper-parameters into array
    end for
until Data is converted to flat 2d dataset
Split Training and Test based on UnitToDrop
repeat
    Reshape Training Dataset
    for i ← 1, rows do
        Reshape Training dataset to 3 Dimension
        Reshape Test dataset to 3 Dimension
    end for
until Training and Test datasets are reshaped
Return trainingDataset, testDatasetData
```

To simplify this process, we used the *Keras* library with a wrapper API around *Theano* and *Tensorflow*. The Keras one_hot function was used to convert strings into indices, form a 2-dimension list and create a dictionary at the same time. Finally, since deep neural networks require arrays to be of equal length, we needed to find the maximum length of a sentence within the *Info* feature and pad all the arrays with zeros to be equal to the maximum length of 25.

After processing the dataset it was split into *training* and *test* datasets and reshaped into 3 dimensions, the format required for LSTM layer (see Algorithm 1.)

B. Machine Learning Module

Attack detections in IoT systems is notably different from the existing mechanisms because of the special service requirements, such as low latency, resource limitations, distributed nature, mobility, to mention a few [7]. This means that conventional network attack detection has limited application in addressing IoT security problems. At the same time, it is known that a considerable number of zero-day attacks are continuously emerging because of the addition of various IoT protocols. Most of these attacks are small variants of previously known cyber-attacks that present a difficulty in their detection even for advanced computational intelligence mechanisms such as traditional machine learning systems.

On the other hand, the success of deep learning (DL) in various big data fields has attracted a noticeable interest in cybersecurity fields. The application of DL has become practical because of the advances in computer architecture (e.g. NVIDIA DGX platforms) and in development of new neural network libraries (such as Theano and Tensorflow for instance); also, the availability of large and diverse training datasets made a contribution to the effectiveness of deep learning algorithms. The use of DL for attack detection in IoT systems could be a resilient mechanism to small mutations or novel attacks because of its high-level feature extraction capability. The self-taught and compression capabilities of deep learning architectures are key mechanisms for hidden pattern discovery from the training data so that attacks are discriminated from benign traffic [21]. This is very important in the context of IoT security because such systems face a plethora of security problems, including jamming, spoofing, replaying and eavesdropping, but also prone to issues related to resource constraints e.g. out-of-memory accesses, unsafe programming languages, etc. [22].

This research is aimed at adopting a deep learning approach to cybersecurity to enable the detection of botnet attacks. Other machine learning and evolutionary computing techniques have been successfully applied in mitigating against botnet attacks. One example is the use of swarm intelligence for destroying any rigid master-slave relationship between bots and for autonomizing the bot operating roles [23]. We will focus on deep learning, however, due to a number of advantages provided by this approach, including the absence of manual feature engineering, unsupervised pre-training, and compression capabilities, which makes the application of deep learning feasible even in resource-constrained networks.

One implication of observing the network traffic over a long period is the necessity to successfully deal with large data sequences. Recurrent neural networks (RNN) in general, and one of its variants the Long Short Term Memory (LSTM) network have been proven effective in recognizing the different sequences of states that change over time [24].

An LSTM network has a recurrent architecture and is able to bridge long time lags between relevant input and target output. This type of structure is theoretically well suited and has been proven a powerful model for tagging tasks with applications in natural language processing, machine translation, Image recognition, and the like [20]. The main contribution of this paper is the application of LSTM networks for implementing deep learning in network traffic analysis aimed at detecting botnet attacks.

IV. Results and Discussion

To evaluate our detection model we required a dataset which contained a mixture of IoT botnet communication, multiple attack vectors and normal IoT device traffic. There are currently no public datasets that fulfilled all three criteria, therefore an experimental set-up was implemented as described in Section III. The *mirai* botnet malware contains ten available attack vectors, which infected IoT devices can utilise to engage in DDoS attacks against targets. For the purpose of our experiments, four attack vectors were chosen, including User Datagram Protocol *(UDP)* flood, Synchronize *(SYN)* flood, Acknowledgement *(ACK)* flood, and Domain Name System *(DNS)* Flood attacks. Command and control messages between the C&C server and the infected IoT IP camera *(bot)* were also captured, as was normal traffic generated by the camera.

To capture packets and generate the necessary dataset the tcpdump command *tcpdump W 5 C 500 w datacapture* was issued, where *-W* stipulates to split the capture into a maximum of five files and *-C* stipulates that the maximum capture file size should be 500MB.

The necessary data was captured in a series of five separate captures, which would later be concatenated into a single dataset. The first capture *(normal.pcap)* consisted of normal IoT device traffic, for a duration of 1 hour and included normal device communication on the network, and also two remote connections to the camera to view the video feed, each of which lasted 5 minutes.

The initial scanning process and device infection was captured in the second capture *(mirai.pcap)* which also included the infected camera scanning on ports 23 and 2323 for new devices to infect. The third capture *(udp.pcap)* consisted of a single *(udp)* flood attack, whereby the C&C server issued the attack command, and the infected IoT device flooded its target with bursts of *(udp)* packets for a total period of 60 seconds. This procedure was repeated for the fourth *(dns.pcap)* and fifth *(ack.pcap)* captures, capturing bursts of *(dns)* and *(ack)* packets respectively.

After capturing all five attack scenarios using the *.pcap* format, the capture files were converted to *.csv* files. In order to train and validate our detection model, ground-truth labels *norm*, *mirai*, *udp*, *dns*, *ack* were assigned to the captured data, ready to be ingested into the detection model. The total number of samples captured by each attack type can be seen in Table II. The cleaned column represents the total number of samples once packets with missing data have been removed.

For *Experiment 1* each attack type was split between *train* and *validate*, presented and trained over a total of 30 iterations. The resulting accuracy and loss metrics for each attack were measured, and the process repeated through ten times. The mean value for each metric is presented in Table III. As can be seen from the results the model returned high accuracy and prediction for *mirai*, *udp*, and *dns* attack types. However, returned less favourable results for *ack* attacks, despite this attack having the hightest number of samples. This was possibly due to the nature and complexity of information in the *info* feature, as seen in Table I, where the sequence number in each *ack* packet changed. By contrast, although the *mirai* captured packets appear to be equally complex, the information in the *info* feature, remained largely the same, possibly aiding better detection.

Since multi-vector DDoS attacks were highlighted as being a growing issue in Section II-A, *Experiment 2* consisted of *norm*, *mirai*, *udp*, *dns*, and *ack* captures being concatenated to form a multi-vector attack scenario. Results on row 1 of Table IV show the impact of the *ack* attack on the overall detection accuracy and prediction metrics. To validate this observation, *Experiment 3* consisted of *norm*, *mirai*, *udp*, and *dns* captures being concatenated to form a multi-vector attack scenario, minus the *ack* attack. Results on row 2 of Table IV show that once the *ack* attack is removed, overall detection accuracy and prediction of the model are very good. A final validation of this observation was conducted in *Experiment 4* which consisted of two *ack* attacks performed during the same time frame, increasing the total sample size of *ack* attacks, in order to observe the variation in accuracy and prediction.

As can be seen on row 3 of Table IV an increase in sample size, significantly improves the overall validation accuracy to 94% and validation loss to 0.03%, meaning the model was able to better predict attack traffic.

The variance in accuracy and prediction possibly indicated that the model required a larger sample size for the *ack* attack type, for it to be effective in detecting and predicting these types of attacks. Fig. 4 and Fig. 3 show the results of multi-vector attack with two iterations of *ack* attacks.

V. Conclusions and Future Work

This paper presents the implementation of a detection model based on a Deep Bidirectional Long Short Term Memory Recurrent Neural Network (BLSTM-RNN), in conjunction with Word Embedding, to convert string data into a format usable by the BLSTM-RNN. In doing so, a public IoT botnet dataset was also created and used to test and validate the designed detection model.

The aim of this paper, was to present an application of deep learning to IoT security, and address the problem stated in Section I, whereby increasingly consumer level IoT devices are being infected and leveraged to perform large scale DDoS attacks against targets on the Internet. This problem is further compounded due to the lack of technical knowledge or situational awareness of inherent vulnerabilities, by owners of consumer IoT devices, meaning it is practically impossible for them to detect and be aware when their device is infected and part of a botnet. The proposed model addresses the issue of detection which returns high accuracy and low loss metrics for four attack vectors used by the *mirai* botnet malware. Results for *mirai*, *udp*, and *dns* were very encouraging with 99%, 98%, 97% validation accuracy and 0.13% 0.24%, 0.23% validation loss metrics respectively. The *ack* attack vector metrics were shown to be less favourable, but the paper showed that a larger sample size could significantly improve accuracy.

TABLE I. Sample Captured Packets

Packet	Time	Source	Destination	Protocol	Size	Info
Normal	0.000226	192.168.252.40	192.168.252.60	TCP	66	81 - 50451 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1 WS=2
Mirai	0.268276	192.168.252.40	106.65.144.6	TCP	64	62002 - 23 [SYN] Seq=0 Win=57378 Len=0 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
UDP	0.268276	192.168.252.40	192.168.252.50	UDP	554	55741 - 65170 Len=512
ACK	1.991338	192.168.252.40	106.65.144.6	TCP	566	21658 - 24366 [ACK] Seq=1 Ack=1 Win=29597 Len=512
DNS	4.513663	192.168.252.40	192.168.252.22	DNS	90	Standard query 0x0c9 A nnt1heibflkk.report.McDPhD.org

TABLE II. Dataset Samples by Captured Attack type

	Attack	Normal	Mirai	Total	Cleaned
Mirai	0	299338	2551	301889	301802
UDP	4690	293762	1288	299740	299740
ACK	33722	294280	3186	331188	331097
DNS	4353	294205	2204	300762	300694

TABLE III. Accuracy of Detection by single Attack type

	Train	Validate	Mean Acc	Mean Loss
Mirai	196171	105631	99.154744	0.1362400
UDP	194831	104909	98.005605	0.2439042
ACK	215213	115884	88.852511	1.6414504
DNS	195451	105243	97.819378	0.2333340

TABLE IV. Accuracy of Detection by Multi-vector Attack types

	Train	Validate	Mean Acc	Mean Loss
Multi-vector (with ACK)	228998	123307	85.241678	1.5425814
Multi-vector (without ACK)	204674	204674	95.209029	0.2228190
Multi-vector (with two ACK)	253321	253321	93.899201	0.0384694

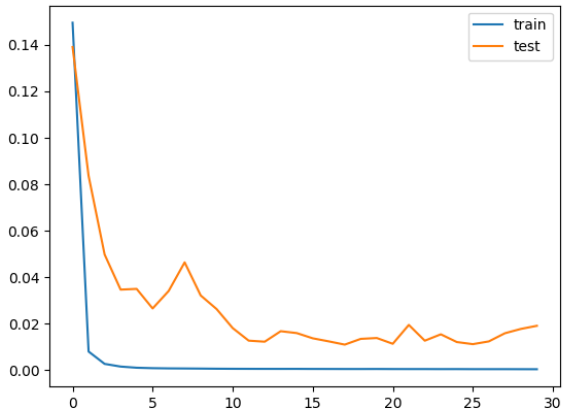

Fig. 3. Model Loss

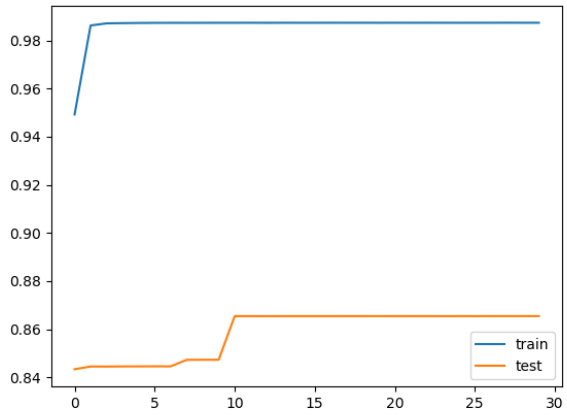

Fig. 4. Model Accuracy

In the future, we plan to create a second more extensive dataset, that will span a longer time frame, and include multiple iterations of each attack vector. In addition, we plan to continue this research and develop the presented detection model, into a full detection engine, capable of ingesting live data feeds, and performing anomaly detection for future mutations of IoT botnets. The second part of the presented problem in Section I, will also be further investigated to explore ways of communicating to the owner, when their consumer level IoT device is infected, and part of a botnet. By helping consumers become more situationally aware when their devices are infected, we hope to raise awareness of the inherent vulnerabilities, and aid them to make better choices in the future, with regard to procurement, and operation of such devices.

The generated *mirai* botnet dataset has been made public and is available upon request.