Botnet detection in the Internet of Things using deep learning approaches.

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

2018

Botnet Detection in the Internet of Things using Deep Learning Approaches

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

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

***Abstract*—The recent growth of the Internet of Things (IoT) has resulted in a rise in IoT based DDoS attacks. This paper presents a solution to the detection of botnet activity within consumer IoT devices and networks. A novel application of Deep Learning is used to develop a detection model based on a Bidirectional Long Short Term Memory based Recurrent Neural Network (BLSTM-RNN). Word Embedding is used for text recognition and conversion of attack packets into tokenised integer format. The developed BLSTM-RNN detection model is compared to a LSTM-RNN for detecting four attack vectors used by the *mirai* botnet, and evaluated for accuracy and loss. The paper demonstrates that although the bidirectional approach adds overhead to each epoch and increases processing time, it proves to be a better progressive model over time. A labelled dataset was generated as part of this research, and is available upon request.**

***Index Terms*—Deep Learning, LSTM, Word Embedding, IoT, Botnet, Mirai, DDoS.**

I. Introduction

The Internet of Things (IoT) is expected to usher in an era of increased connectivity, with an estimated 50 billion devices expected to be connected to the Internet by 2020 [1]. At its core, the aim of the IoT is to connect previously unconnected devices to the Internet [2], thus creating smart devices capable of collecting, storing and sharing data, without requiring human interaction [3] [4]. Many of these IoT devices are aimed at consumers, who value low cost and ease of deployment over security. These market forces have resulted in IoT manufacturers omitting critical security features, and producing swathes of insecure Internet connected devices, such as IP cameras and Digital Video Recorder (DVR) boxes. Such vulnerabilities and exploits are often derived and epitomised by inherent computational limitations, use of default credentials and insecure protocols. The rapid proliferation of insecure IoT devices and ease by which attackers can locate them using online services, such as shodan, provides an ever expanding pool of attack resources. By comprising and leveraging multitudes of these vulnerable IoT devices, attackers can now perform large scale attacks such as spamming, phishing and Distributed Denial of Service (DDoS), against resources on the Internet [5].

The rise in IoT based DDoS attacks, witnessed in recent years, will likely continue until IoT manufacturers accept responsibility and incorporate security mechanisms into their devices. Until such a time, the IoT has the potential to become the new playground for future cyber attacks and therefore presents a number of challenges. Since an increasing number of DDoS attacks seek to leverage consumer level IoT devices, the issues highlighted previously, coupled with a lack of technical knowledge or awareness of inherent vulnerabilities, by owners of these devices, presents one such problem. This challenge is further compounded by a lack of convenient user interface on many consumer IoT devices, making detection and awareness of attacks in home networks practically impossible for consumers.

To substantiate this issue, we undertook preliminary research and created a secure sandboxed botnet environment. An IoT IP Camera was successfully infected, and leveraged to perform a sequence of DDoS attacks against a selected target. During the infection process and attacks, the camera did not display any adverse symptoms of infection, and continued to function as expected. Remote access to the device was still possible, and performance did not appear to be degraded. Live video streaming continued to be as responsiveness as prior to the attacks, therefore without any clear signs of an infection it was confirmed that, detection or awareness or botnet activity would prove very difficult within consumer networks.

Current methods of botnet detection such as signature or flow based anomaly intrusion detection, have proved ineffective in preventing the spread of IoT botnets. Largely due to simple code mutations rendering attack signatures obsolete or a lack of protocol support (*NetFlow, Sflow*) within consumer networks and equipment.

This paper presents a solution to the detection of botnet activity within consumer IoT devices and networks. A novel detection model was developed based on a Deep Bidirectional Long Short Term Memory based Recurrent Neural Network (BLSTM-RNN). Detection was performed at the packet level, and focused on text recognition within features, normally discarded by other flow based detection methods. Word Embedding was used for text recognition and conversion, and proved to be an effective method for predicting attack vectors. The BLSTM-RNN detection model was compared with a LSTM-RNN, and evaluated for accuracy and loss.

The main contributions of this paper can be defined as:

1) Producing a labelled and public dataset incorporating botnet traffic, attack vectors, and normal traffic;
2) Developing a detection algorithm for text recognition of features within botnet attack vectors;
3) Comparing LSTM and BLSTM Recurrent Neural Network based detection models to detect and predict infected IoT device traffic.

The rest of the paper is organized as follows: Section II introduces botnet activity within the IoT, and the application of deep learning for attack detection. Section III describes the botnet architecture used to generate the botnet dataset. It also details the use of a BLSTM-RNN in conjuction with Word Embedding methodology to create a botnet detection model. Section IV describes the process of data collection and pre-processing. Section V evaluates the experimental results, comparing the LSTM and BLSTM Recurrent Neural Network models for accuracy and loss. Section VI draws conclusions and suggests possible future research directions.

II. Security 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. During this time, attacks of over 100 Gbps, were up by 140% with three attacks reaching over 300 Gbps [6]. Fuelled in full or part by the Internet of Things, 88% of DDoS attacks in quarter 4 of 2017 employed a multi-vector attack strategy [7].

A. Botnets in the Internet of Things

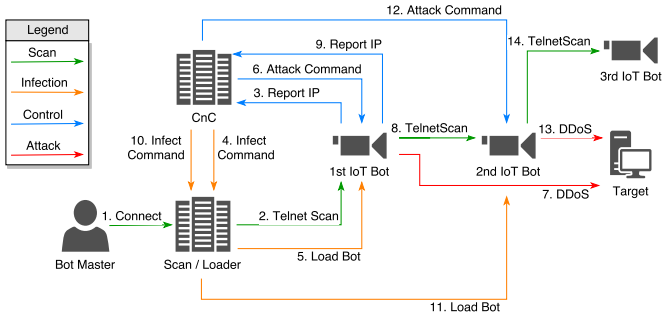

Fig. 1. Botnet Infection and Proliferation

One of the most prominent examples of a DDoS attack emanating from the IoT during this period was the Mirai botnet. 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 [8]. 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 [9].

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.

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 (*step 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. Deep Learning for Attack Detection

The increasing presence of IoT systems in a broad range of applications, as well as their increasing computing and processing capabilities make them a valuable attack target, such as network packets and malware designed to compromise specific IoT devices. Attack detections in IoT systems is notably different from the existing mechanisms because of the special service requirements, such as low latency, resource specificity, distributed nature, mobility, to mention a few [10]. This means that conventional network attack detection has limited application in addressing IoT security problems. According to Kaspersky Lab, in 2016 the majority of IoT devices examined were insecure, using default passwords or unpatched vulnerabilities, and easily compromised by Mirai and Hajime malware [11].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.

Previous literature have suggested the potential of leveraging machine learning to enhance security threat hunting, but it is not practical to simply integrate machine learning in static and dynamic cyber security analysis due to the wide variety and distribution of IoT devices, particularly for (inexpensive) IoT devices with limited processing power [12]. On the other hand, the success of deep learning (DL) in various big data fields has attracted 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.

Deep learning (DL) enables several breakthroughs of conventional AI tasks in the fields of image processing, pattern recognition and computer vision. Deep networks are capable of achieving significant improvement in accuracy of classification and predictions in these complex tasks. The main benefit of deep learning is the absence of manual feature engineering, unsupervised pre-training and compression capabilities which enable the application of deep learning feasible even in resource constraint networks. It means that the capability of DL to self-learning results in higher accuracy and faster processing, which can be effectively utilised for a novel distributed attack detection in IoT systems [13]. 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. [14].

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 [15]. The evolving behaviour of botnets often enables them to circumvent the traditional detection approaches. The development of behavioural detection approaches, however, have helped in dealing with the constant change in the botnet activities by finding the common patterns that botnets follow across their life cycle. For instance, all the bots need connect to the C&C server to receive new orders, and this kind of behaviour observed only after a long period of time can guide the detection methods.

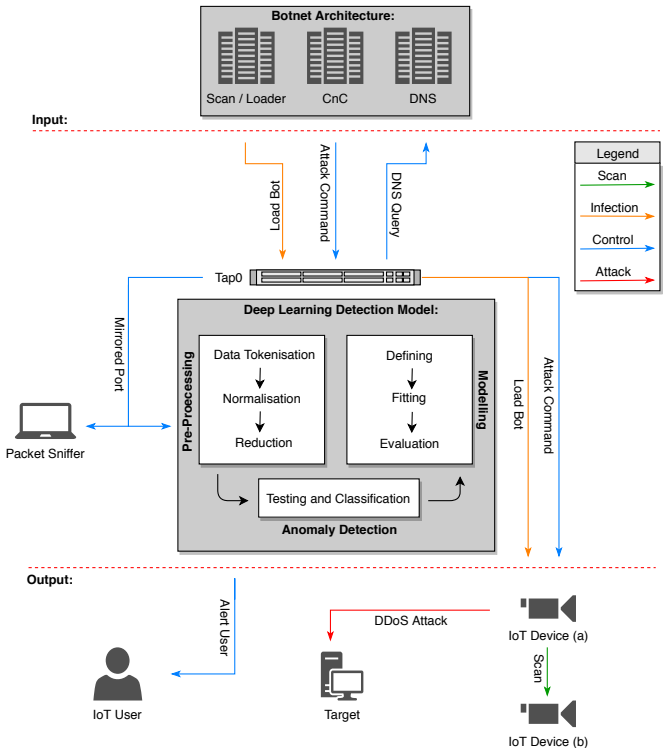

Fig. 2. Botnet Architecture and Deep Learning Detection Model

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, bridging thereby long time lags between relevant input and target output [16]. 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 [17]. A bidirectional LSTM (BLSTM), furthermore, introduces two independent layers to accumulate contextual information both from the past and the future [12]. The main contribution of this paper is the application of the variants of LSTM networks for implementing deep learning in network traffic analysis aimed at detecting botnet attacks.

III. Methodology

To promote reproducibility of this paper, a detailed description of botnet environment and algorithm implementation is presented.

A. Experimental Setup

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.

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*.

Algorithm 1 Botnet Detection Algorithm

```
dataProcessing (dataset)
unitToDrop ← 25%
Parse data to predefined format
Define token dictionary
repeat
    /*Parse data to format*/
    for row ← 1, rows do
        Convert text to tokenised integer format
        Index tokenised text
        Create dictionary of tokenised text indices
        Pad data arrays with 0s to max 25
        Inject additional tokenised features into array
    end for
until return dataset
Split Training and Test based on unitToDrop
TrainAndValidate (trainingData, testData)
model ← sequential()
cell ← 0
activation ← sigmoid
loss ← mae
optimiser ← Adam
epochs ← 100
Create new BLSTM/LSTM unit
Add LSTM unit to model
Create new Dense Layer
Add Dense Layer to model
Set activation for Dense Layer
Compile model using Optimiser and Loss
repeat
    /*Fit Model*/
    for epoch ← 1, epochs do
        Evaluate Loss, Validation Loss
        Evaluate Accuracy and Validation Accuracy
    end for
until All epochs completed
Return Loss, ValLoss, Acc, ValAcc
```

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 delivering 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.

B. Pre-Processing using Word Embedding

The developed 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.

The dataset used in our experiments was generated from the experimental set-up described in Section III-A. 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 III therefore a model was required which could read and understand the text presented in this feature. As discussed in Section III-C 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 [18] 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 [17] 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 detection 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. 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 0 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.)

TABLE I. Model Parameters

Variables	**Values**
Activation	Sigmoid
Loss	Mean Absolute Error (mae)
Optimiser	Adam
BLSTM layer total units	20
Dense layer total unit	6
Epochs	100

TABLE II. Captured Attack Samples

	Attack	**Normal**	**Mirai**	**Cleaned**
Mirai	0	598676	5102	595478
UDP	9380	587524	2576	601542
ACK	67444	588560	6372	632889
DNS	8706	588410	4408	602496

C. Modelling using Long Short Term Memory Recurrent Neural Network

As previously stated the main contribution of this paper is the application of deep learning to botnet detection in the IoT. Word embedding was used in Section III-B to convert text into tokenised integer format, for use with a deep neural network. To test the effectiveness of this approach the detection model is evaluated using LSTM-RNN and BLSTM-RNN and tested against a series of attacks associated with the *Mirai* botnet. As shown in algorithm 1 to develop the detection models, unit and Output layer with sigmoid activation are added to the model. The model is then compiled with the MAE loss function and the Adam optimiser over total of 100 iterations, as shown in Table I.

The proposed detection model shown in Fig. 2 transitions acquired botnet data through three distinct phases. The Preprocessing phase adjusts features to ensure data representation is suitable for the used algorithms. The Word Embedding method described in Section III-B tokenises the data, before normalisation and removal of packets with missing data.

In the Modelling phase the LSTMN-RNN and BLSTM-RNN algorithms are applied to the training data to define, fit and evaluate the detection model.

Finally in the Anomaly Detection phase the generated dataset is tested to determine the effectiveness of the model in terms of accuracy and loss.

IV. Data Sources

To evaluate our detection models 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-A. 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, Acknowledgement *(ACK)* flood, Domain Name System *(DNS)* flood, and Synchronize *(SYN)* flood attacks, used by *mirai*. Command

TABLE III. Attack Packet Structure

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
DNS	4.513663	192.168.252.40	192.168.252.22	DNS	90	Standard query 0x0c9 A nnt1heibflkk.report.McDPhD.org

TABLE IV. ACK Packet Structure and Sequencing

Packet	Time	Source	Destination	Protocol	Size	Info
ACK	1.940214	192.168.252.40	192.168.252.50	TCP	566	59693 - 41058 [ACK] Seq=1 Ack=1 Win=29597 Len=512
ACK	1.940431	192.168.252.50	192.168.252.40	TCP	60	41058 - 59693 [ACK] Seq=1 Ack=1 Win=29597 Len=0
ACK	1.959063	192.168.252.40	192.168.252.50	TCP	566	28029 - 45060 [ACK] Seq=1 Ack=1 Win=29597 Len=512
ACK	1.959074	192.168.252.40	192.168.252.50	TCP	566	56493 - 64047 [ACK] Seq=1 Ack=1 Win=29597 Len=512

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 2 hours 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.

Mobaxterm was used to create a secure shell *(ssh)* into the C&C server, before executing command *screen ./cnc* from within the *mirai/release* directory, to start the *MYSQL* database. A second remote session was used to telnet and log into the C&C server, ready to issue attack commands to the infected IoT IP camera. A third remote session was used to ssh into the Scan/Loader server, before executing the *./loader* command from within the *mirai/release* directory, to scan the network for available IoT devices to infect.

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. The fourth capture *(dns.pcap)* consisted of a single *(dns)* flood attack, whereby the C&C server issued the attack command, and the infected IoT device flooded its target with bursts of *(dns)* packets for a total period of 60 seconds. The fifth capture *(ack.pcap)* consisted of a single *(ack)* flood attack, whereby the C&C server issued the attack command, and the infected IoT device flooded its target with bursts of *(ack)* packets for a total period of 60 seconds.

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.

V. Model Comparison and Discussion

To compare our deep learning detection models a series of four experiments were performed for each. Since unidirectional LSTM-RNN only preserve information from the past, the aim of the comparison was to ascertain if the use of a bidirectional LSTM-RNN, which is able to accumulate contextual information from both past and future, could return better accuracy or loss metrics for our captured dataset. For *Experiment 1* each attack type was split between *train* and *validate*, presented to each model and trained over a total of 20 iterations. The mean accuracy and loss metrics for each attack were measured, and are presented in Table V. As can be seen from the results, both models 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 highest number of samples. This was possibly due to the nature and complexity of information in the *info* feature, as seen in Table IV, where the sequence numbers in each *ack* packet changed. Despite this, a pattern can however be seen on rows one and two, where sequence numbers *(59693-41058, 41058-59693)* of contiguous packets were clearly linked, and packet size and Length were consistent. Unfortunately some packets appeared out of sync as can been in rows three and four, and possibly resulted in the detection model not recognising this pattern, contributing to the lower detection rate, and significantly higher loss metric. By contrast, although the *mirai* captured packets in Table III 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, *Experiment 2* consisted of *norm*,

TABLE V. Detection Accuracy and Loss

	Train	Validate	BLSTM Accuracy	LSTM Accuracy	BLSTM Loss	LSTM Loss
Mirai	387060	208418	99.998992	99.571605	0.000809	0.027775
UDP	391002	210540	98.582144	98.521440	0.125630	0.125667
ACK	411384	221515	93.765198	93.765198	0.858700	0.858773
DNS	391622	210874	98.488289	98.488289	0.116453	0.116453
Mulit-Vector (with ACK)	419887	226094	91.951002	91.951002	0.841303	0.841381
Mulit-Vector (without ACK)	395564	212996	97.521033	97.521033	0.115293	0.115293
Mulit-Vector (with three ACK)	468534	252289	92.243513	92.243513	0.161890	0.242358

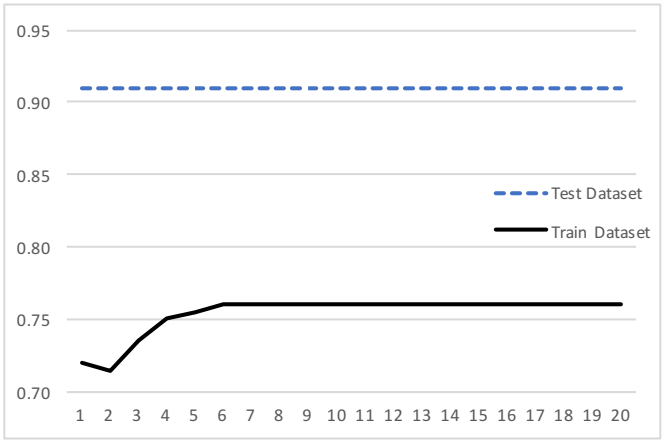

Fig. 3. LSTM Accuracy

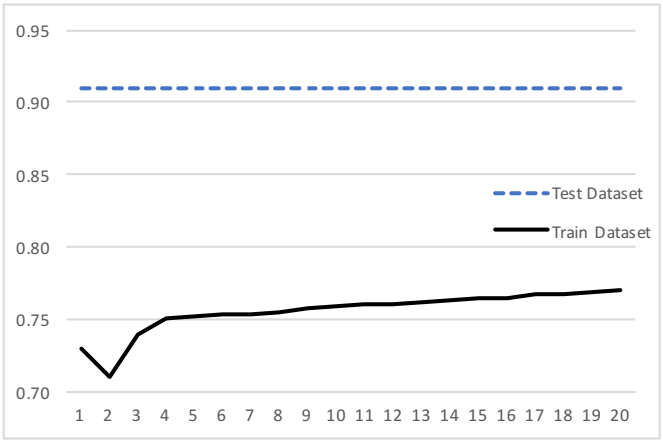

Fig. 5. BLSTM Accuracy

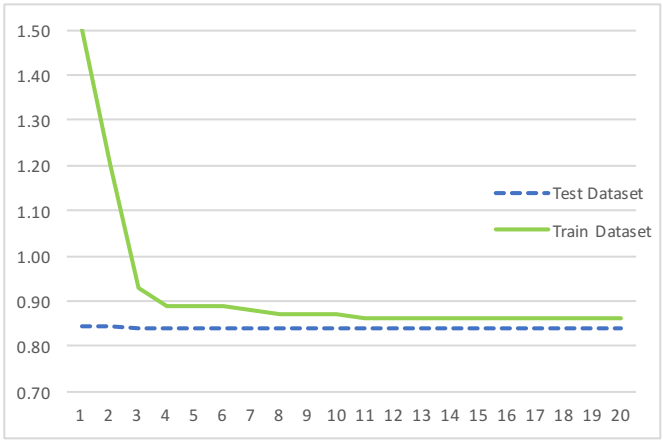

Fig. 4. LSTM Loss

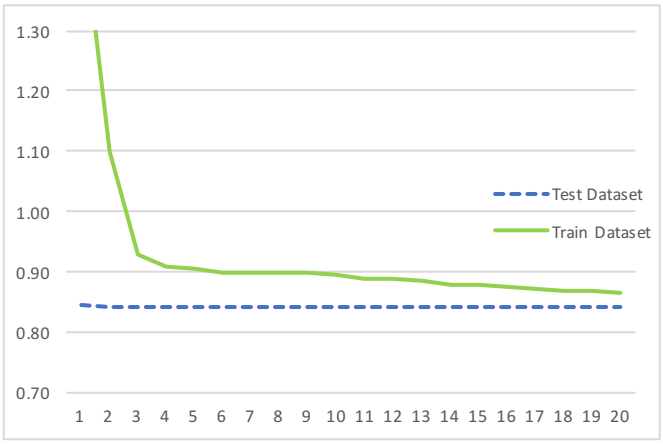

Fig. 6. BLSTM Loss

mirai, *udp*, *dns*, and *ack* captures being concatenated to form a multi-vector attack scenario. Results on row 5 of Table V show the impact of the *ack* attack on the overall detection accuracy and particularly loss 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 6 of Table V 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 three *ack* attacks were performed during the same time frame, increasing the total sample size of *ack* attacks, in order to observe the variation in accuracy and prediction.

Row 7 of Table V shows an increase in sample size, improves the overall validation accuracy to 92%, with BLSTM-RNN returning the better loss metric, meaning this model was able to better predict attack traffic, when presented with a larger sample size.

Fig. 3 through to Fig. 6 show accuracy and loss metrics for the detection models. Although metric results are comparable, and the bidirectional approach adds overhead to each epoch, increases processing time, the trajectory shows a better progressive model over time. A larger dataset with more samples, could further demonstrate the benefit of BLSTM-RNN.

VI. Conclusions and Future Work

This paper presents the implementation of deep learning using a Bidirectional Long Short Term Memory Recurrent Neural Network (BLSTM-RNN), in conjunction with Word Embedding for botnet detection. The model was compared to a unidirectional LSTM-RNN to ascertain if the accumulation of contextual information from both past and future used by the BLSTM-RNN, could return better accuracy or loss metrics for our captured dataset. Both models returned high accuracy and low loss metrics for the four attack vectors used by the *mirai* botnet malware. Results for *mirai*, *udp*, and *dns* were very encouraging with 99%, 98%, 98% validation accuracy and 0.000809, 0.125630, 0.116453 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 improve accuracy and reduce loss. The positive results demonstrate the effectiveness of our novel application of deep learning for botnet detection in the IoT. By focusing detection at the packet level, and using text recognition on features normally discarded, we have demonstrated that the limitations of existing specification or flow based detection methods, can be overcome. Furthermore, although the bidirectional approach adds overhead to each epoch, and increases processing time, it appears to be a better progressive model over time

Several avenues for future research have been identified. Firstly a second more comprehensive dataset will be generated, incorporating all ten attack vectors used by the *mirai* botnet malware. To demonstrate the ability of our developed model to detect new variations of botnets, a mutated version of the *mirai* source code will be used to generate a third dataset, and will be compared against existing signature and flow based anomaly detection methods. Having successfully demonstrated a solution to the detection problem presented in Section I, we will also further investigate ways to improve situational awareness of botnet activity within the IoT. By helping consumers become aware when their device is 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.

References

[1] M. Aazam, M. St-Hilaire, C.-H. Lung, I. Lambadaris, and E.-N. Huh, *IoT Resource Estimation Challenges and Modeling in Fog*. Springer International Publishing, 2018, pp. 17–31.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[3] A. Mosenia and N. K. Jha, "A comprehensive study of security of internet-of-things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586–602, Oct 2017.

[4] C. D. McDermott and A. V. Petrovski, "Investigation of computational intelligence techniques for intrusion detection in wireless sensor networks," *International Journal of Computer Networks and Communications (IJCNC)*, vol. 9, no. 4, pp. 45–56, 2017.

[5] S. Moganedi and J. Mtsweni, "Beyond the convenience of the internet of things: Security and privacy concerns," in *2017 IST-Africa Week Conference (IST-Africa)*, May 2017, pp. 1–10.

[6] Akami. (2017) Threat Advisory Internet of Things and the Rise of 300 Gbps DDoS Attacks. [Online]. Available: https://www.akamai.com

[7] Verisign, "Verisign distributed denial of service trends report," *Computer Networks*, vol. 4, 2017.

[8] B. Krebs. (2016) KrebsOnSecurity Hit With Record DDoS. [Online]. Available: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/

[9] S. Hilton. (2016) The DDoS that didn't break the camel's VAC. [Online]. Available: https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/

[10] J. A. Jerkins, "Motivating a market or regulatory solution to iot insecurity with the mirai botnet code," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2017, pp. 1–5.

[11] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[12] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computer Systems*, vol. 85, pp. 88 – 96, 2018.

[13] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Generation Computer Systems*, pp. 1–5, 2017.

[14] F. A. Teixeira, F. M. Pereira, H.-C. Wong, J. M. Nogueira, and L. B. Oliveira, "SIoT: Securing Internet of Things through distributed systems analysis," *Future Generation Computer Systems*, 2017.

[15] A. Castiglione, R. D. Prisco, A. D. Santis, U. Fiore, and F. Palmieri, "A botnet-based command and control approach relying on swarm intelligence," *Journal of Network and Computer Applications*, vol. 38, pp. 22 – 33, 2014.

[16] P. Torres, C. Catania, S. Garcia, and C. G. Garino, "An analysis of Recurrent Neural Networks for Botnet detection behavior," in *2016 IEEE Biennial Congress of Argentina (ARGENCON)*, June 2016, pp. 1–6.

[17] P. Wang, Y. Qian, F. Soong, L. He, and H. Zhao, "A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding," *ArXiv e-prints*, Nov 2015.

[18] A. Ray, S. Rajeswar, and S. Chaudhury, "Text recognition using deep blstm networks," in *2015 Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, Jan 2015, pp. 1–6.