Journal Pre-proof

An efficient reinforcement learning-based Botnet detection approach

Mohammad Alauthman, Nauman Aslam, Mouhammd Alkasassbeh, Suleman Khan, Ahmad AL-qerem, Kim-Kwang Raymond Choo

PII: S1084-8045(19)30339-X

DOI: https://doi.org/10.1016/j.jnca.2019.102479

Reference: YJNCA 102479

To appear in: *Journal of Network and Computer Applications*

Received Date: 2 May 2019

Revised Date: 20 September 2019

Accepted Date: 27 October 2019

Please cite this article as: Alauthman, M., Aslam, N., Alkasassbeh, M., Khan, S., AL-qerem, A., Raymond Choo, K.-K., An efficient reinforcement learning-based Botnet detection approach, *Journal of Network and Computer Applications* (2019), doi: https://doi.org/10.1016/j.jnca.2019.102479.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2019 Published by Elsevier Ltd.

Journal Pre-proof

An Efficient Reinforcement Learning-Based Botnet Detection approach

Mohammad Alauthman[a,*], Nauman Aslam[b], Mouhammd Alkasassbeh[c], Suleman Khan[b], Ahmad AL-qerem[c], Kim-Kwang Raymond Choo[d]

[a]*Department of Computer Science, Faculty of information technology, Zarqa University, Zarqa, Jordan*
[b]*Department of Computer Science and Digital Technologies, Faculty of Engineering and Environment, Northumbria University, Newcastle upon Tyne, UK, NE1-8ST*
[c]*Department of Computer Science, Princess Sumaya University for Technology, Amman, Jordan*
[d]*Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249, USA*

Abstract

The use of bot malware and botnets as a tool to facilitate other malicious cyber activities (e.g. distributed denial of service attacks, dissemination of malware and spam, and click fraud). However, detection of botnets, particularly peer-to-peer (P2P) botnets, is challenging. Hence, in this paper we propose a sophisticated traffic reduction mechanism, integrated with a reinforcement learning technique. We then evaluate the proposed approach using real-world network traffic, and achieve a detection rate of 98.3%. The approach also achieves a relatively low false positive rate (i.e. 0.012%).

Keywords: Botnet detection, Network security, Traffic reduction, Neural network, C2C, Reinforcement-learning.

*Corresponding author
Email addresses: malauthman@zu.edu.jo (Mohammad Alauthman), nauman.aslam@northumbria.ac.uk (Nauman Aslam), m.alkasassbeh@psut.edu.jo (Mouhammd Alkasassbeh), suleman.khan@northumbria.ac.uk (Suleman Khan), a.qerem@psut.edu.jo (Ahmad AL-qerem), raymond.Choo@fulbrightmail.org (Kim-Kwang Raymond Choo)

Journal Pre-proof

1. Introduction

Bot malware and botnets are two widely understood concepts in the cyber security literature. Specifically, a botnet is a network of geographically dispersed infected bots (e.g. any computing device including an Internet of Things (IoT) device, such as a smart TV, that has been compromised by a bot malware), which is remotely controlled by a botmaster. Such botnets are generally used to carry out a range of malicious cyber activities, ranging from sending of spams to launching of distributed denial of service (DDoS) attacks to dissemination of malicious programs (malware) to disseminating illegal materials (e.g. child exploitation materials) to click fraud, and so on [1, 2, 3, 4].The communication channel between the botnet and the botmaster is also referred to as the command and control (C2C) channel, which can be either centralized or decentralized [5, 6, 7, 8, 9]. Decentralized C2C infrastructures, such as peer-to-peer (P2P) infrastructure, are generally harder to detect in comparison to centralized C2C infrastructures. This is also partly evidenced by the increased adoption of the P2P infrastructure in botnets [10], such as Waledac Bot [11], Conficker Bot [12], Zeus Bot [13], and Storm Bot [14].

A typical P2P botnet lifecycle comprises four main stages, namely: initial infection, peer discovery, secondary update and attack [10]. In the first phase, the bot malware is installed on an end-user computing device (e.g. Internet of Things device, such as a smart TV, an edge device, and/or an industrial control system), say by exploiting known vulnerabilities or using social engineering (e.g. via email attachments, drive-by-downloads) [15, 16, 17]. This is also done without the victim's knowledge. During the second phase, the bot will seek to establish a connection with other bots (i.e. infected hosts) that are in the same botnet. In the third phase, the bot attempts to download and install the latest update of the bot malware, if it exists (this is analogous to installing a new version of a mobile app, or patching the system). This phase typically takes place via the C2C channel. In the last phase, the bots will carry out the various malicious cyber activities, on the command of the botmaster.

Journal Pre-proof

There are a number of ways to detect such bots. For example, an organization could analyze their own network traffic and attempt to identify suspicious hosts involved in malicious activity. Existing botnet detection systems, such as those described in [18, 19, 20, 21, 6], generally rely on DPI to analyze the packet contents. This can be computationally expensive and inefficient in recognizing unknown payload signatures. In addition, such detection systems when deployed in high-speed and/or high-volume networks, are generally not capable of performing a comprehensive analysis of all network traffic. Hence, mitigating P2P botnets remains a topic of going interest for both the research community and the practitioner community.

In this paper, we develop an effective reinforcement learning-based detection system, designed to detect and identify infected hosts in a P2P botnet, including new bot (with previously unknown behavior and payload). Specifically, our proposed system comprises a traffic reduction method, in order to deal with a high volume of network traffic. We also attempt to detect the bots as early as possible, for example during the propagation phase (i.e. before the bot launches any malicious activity; in other words, during the earlier discussed peer discovery and secondary update stages). To avoid having a high false positive rate, a set of host traffic features is adaptively set to differentiate between a host infected with a P2P Bot and a legitimate network host.

We will now explain the layout of this paper. In the next section, we will briefly review the relevant literature. In Sections 3 and 4, we present our proposed approach, and describe the evaluation setup and findings. The last section concludes this paper.

2. Related Literature

As discussed in the preceding section, there is an extensive literature on bot malware and botnet detection and it remains a topic of ongoing interest, partly evidenced by the number of research papers [22, 23, 24, 25] and literature review and survey papers on the topic published in recent years [26, 27, 28].

Journal Pre-proof

Botnet detection techniques can be broadly classified into anomaly-based, data mining-based, signature-based and DNS-based techniques [29, 30, 26, 27, 28]. For example, Han et al. [8] classified P2P botnet detection into those based on machine learning, data mining, traffic analysis and network behavior, and Wei et al. classified botnet detection techniques into unsupervised and supervised techniques [31].

In the survey of P2P botnet detection, Babak et al. [32] also proposed a botnet detection system, referred to as PeerRush. The latter employs a one-class classification approach to classify P2P traffic into abnormal traffic and normal traffic. Other techniques used in the classification of abnormal traffic and normal traffic include Gaussian, Parzen, and K-centers data description [33]. Also in [33], the researchers created an application profile based on the analysis ofsome P2P applications' network traffic. In addition, features such as the interval delays between flow duration and packets were used to classify P2P applications. However, such an approcah can be easily circumvented, for example by changing the delay between packets.

Garg et al. [34] studied the potential of using three machine learning algorithms in detecting P2P botnets, namely: J48, Naive Bayes, and Nearest-Neighbor. They aimed to explore the effectiveness of several classifiers. While their studied suggested that both J48 and Nearest-Neighbor achieved reasonably accuracy, the accuracy of detecting legitimate traffic is low. Jiang and Shao [35] proposed relying on the dependency of botnet flows with other peer bots (in the same botnet) to detect bots. Specifically, they used a single-linkage hierarchical clustering mechanism to differentiate between a normal host and a P2P bot. However, it does not detect botnets that utilize irregularity that lies within the traffic flow (e.g. Storm Bot [36]).

Zhang et al. [37] introduced a system to detect hidden P2P botnet, by monitoring the traffic of suspected C2C. The researchers obtained four features from every network flow. Such features include bytes and packets numbers that have been received and sent. The authors used the BIRCH [38] and hierarchical clustering [39] algorithms for clustering network flow. The system showed high

Journal Pre-proof

accuracy rates in detecting malicious and legitimate hosts. It also showed TPR of 100%, and FPR rate of 0.2%. It should be noted that the latter system is capable of detecting botnets regardless of the way the botnets carry out their malicious activities. Despite that, the latter system targets P2P botnets only. This system is criticized for not being capable of detecting other botnet types, such as: the HTTP and IRC bots. In addition, the latter system is vulnerable to several methods of evasion. Such methods include: the flow disturbance packets, DGA and Fast-flux algorithms.

Liao et al. [40] employed a packet size-based methodology for distinguishing between legitimate P2P traffic and P2P botnet traffic. When they evaluated the performance of using Bayesian, J48, and Naive Bayes networks in classifying network traffic, they achieved accuracy rates of 98%, 87%, and 89%, respectively. They also determined that P2P bots' packets size is generally less than normal P2P applications. Similarly, Zhao et al. [41] used REPTree for classification of online P2P botnet detection. However, a key limitation of this approach is that it can be circumvented using random connection interval [42].

The approach of Masud et al. [43] is based on the premise that bots' reaction patterns differ from the reaction patterns of humans. They then demonstrated how one can utilized such an approach to detect bots by identifying the relationship between incoming packets application startups, and outgoing packets and connections. In a separate work, the authors [44] evaluated the potential of using Boosted decision tree, Bayes network classifier, Naive Bayes, support vector machine and C4.5 decision tree in detecting IRC Botnets. It was found that the detection rates for these machine learning techniques are greater than 95 %, with a false positive rate of less than 3% and false negative rate of less than 5%. The Boosted decision tree had the highest overall performance. However, this approach is incapable of detecting botnets that utilized encrypted communication or contemporary bot botnets, such as P2P botnets. More recently in 2018, Wei et al. [31] introduced an unsupervised method based on clustering rather than classification methods. Their approach was not confined to a specific botnet type and is sufficiently flexible.

Journal Pre-proof

There have also been focuses on avoiding detection by existing botnet detection solutions, for example by using encryption [14, 45] or using regular protocols (e.g. P2P and HTTP) [35, 46].

It is clear from these discussed works that botnet, particularly P2P botnet, detection remains an ongoing challenge.

In the next section, we will present our proposed approach.

3. Our Proposed Approach

In the proposed system, we focus on the passive monitoring of network traffic and the frequent communication between bots and their C2C servers during propagation. Specifically, such (frequent) communication is often used to discover other peers and receive commands and related updates [47, 48], and hence can be leveraged to facilitate detection. Our proposed detection approach comprises the following phases: network traffic capture and packet reduction, feature extraction, malicious activity detection, and bot behavior detection using reinforcement learning – see Sections 3.1 to 3.4.

3.1. Network traffic capture and packet reduction

In this phase, network traffic will be sniffed based on the sliding time-window size, and then utilized for traffic reduction. In this paper, we passively capture the network packets using Jpcap [49], since a passive capturing does not (significantly) increase the volume of packets in the network. It also allows us to detect botnets without interacting with them. Given the volume of network packets to be analyzed, network traffic is divided into time-windows. Such a time-window is also required for delivering the results to the network admin on a timely basis. Bots may also seek to generate a temporal behavior after the infection phase [50]. Therefore, using time window can facilitate bot detection. However, we need to determine an appropriate size. For example, if the size is too small, very few captured packets will be captured and hence we are not able to learn the traffic characteristics. If the size of time-window is too large,

Journal Pre-proof

it can be lead to failure in the early detection of botnet behavior. More details on determining time-window size is discussed in Section 4. As shown in Figure 1, $W1, W2...Wn$ denote the sliding time window size.

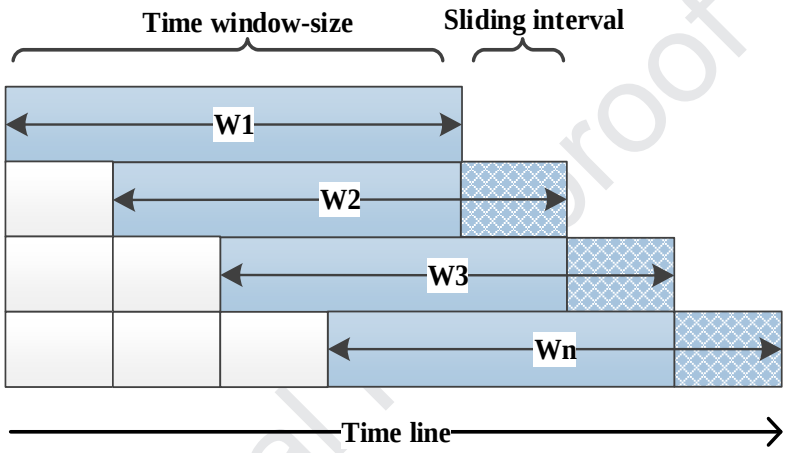

Figure 1: Time-window sliding technique

Given the size of network traffic, we need to efficiently reduce its traffic (i.e. performing a triage). For example, we can use existing approaches such as those described in [51] to reduce the network traffic.

3.2. Feature extraction

We then need to analyze the reduced traffic to identify attributes that can be used to effectively characterize the botnet, and these attributes may collectively form a feature. Clearly, the quality of the features has a significant impact on the detection accuracy of the machine learning algorithm used. Network traffic feature extraction can occur at three levels, namely: packet-level, flow-level, and connection-level [52]. In addition, classification depends on the level of

Journal Pre-proof

packet inspection (e.g. deep or shallow packet inspection). We propose using a mixture of connection and packet levels. For instance, identifying the inter-arrival time features between the packets in each connection requires gathering of packet-level data to be aggregated into connections. That is done for collecting statistical information about connection states. The features employed in our approach are extracted through two stages. First, connection features are extracted. Then, these features serve as host features (representing the host state during the sliding time-window).

3.2.1. Connection level features

This phase focuses on features that are important for the detection of the P2P botnet. In our study, 43 features are collected and gathered in accordance with our pre-determined sliding window size – see received. The features collected comprise control packets exchanged between network hosts and have 5 tuples (IP source address, IP destination address, source port, destination port, protocol).

Journal Pre-proof

Journal Pre-proof

Table 1: Features importance ranking by entropy algorithm

Feature	Description	Feature	Description
F1	# of control packets	F2	# of transmitted control packets
F3	# of received control packets	F4	# of transmitted bytes per flow
F5	# received bytes per flow	F6	# of transmitted SYN packets
F7	# of received SYN packets	F8	# of transmitted ACK packets in a sequence
F9	# of received ACK packets	F10	# of transmitted duplicate ACK packets
F11	# of received duplicate ACK packets	F12	# Avg. length of transmitted control packets
F13	# of received duplicate ACK packets	F14	# Avg. length of transmitted control packets
F15	# of transmitted failed connections	F16	# of received failed connection
F17	# of transmitted SYN-ACK packets/connection	F18	# of received SYN-ACK packets/connection
F19	# of transmitted SYN-ACK in a sequence/connection	F20	# of received SYN-ACK in a sequence/connection
F21	# of bytes per connection/connection	F22	# Ratio of incoming control packets/connection
F23	avg. length of outgoing Ctrl pkt avg. length of Ctrl pkts	F24	Transmitted SYN - received ACK/ Avg. # of SYN
F25	(transmitted SYN - received SYN-ACK)/connection	F26	# of transmitted FIN-ACK packets/connection
F27	# of received FIN-ACK packets per connection	F28	# of transmitted RST-ACK packets per connection
F29	# received RST-ACK packets per connection	F30	avg. time between attempts to create connections
F31	# of received RST packets per connection	F32	of transmitted RST-ACK pkts in a sequence/connection
F33	# of received RST packets per connection	F34	of transmitted RST-ACK pkts in a sequence/connection
F35	Inter-arrival time b/w SYN and ACK by host/connection	F36	Inter-arrival time b/w SYN and RST by host/connection
F37	Inter-arrival time b/w SYN and RST-ACK y host/connection	F38	Inter-arrival time b/w SYN from host RST from other side/connection
F39	Inter-arrival time b/w SYN from host RST-ACK from other side/connection	F40	Inter-arrival time b/w FIN-ACK from host RST from other side/connection
F41	Inter-arrival time b/w ACK from host and connection and RST from other side	F42	Inter-arrival time SYN from host and connection and SYN-ACK from other side
F43	Connection duration		

Journal Pre-proof

Journal Pre-proof

3.2.2. Feature reduction

Feature reduction refers to a method that can minimize the number of attributes in order to eliminate features with minimal impact on the classification problem [53, 54]. The feature reduction technique is adopted for decreasing the 'over-fitting' problem [55], which is crucial in overcoming the imbalanced dataset problem [56]. Thus, the quality of the feature reduction technique is an important factor given its influence over the accuracy rate of the classification algorithm.

We use the classification and regression tree (CART) [57] as the feature reduction technique. The decision tree generated by the CART algorithm has two kinds of nodes, namely: leaf nodes without children and internal nodes with two children. Internal nodes are associated with a decision function to identify the node that shall be visited next. To begin building the tree, training samples along with their class labels are required. During the designing of the tree, the training set is divided recursively into smaller subsets. Through the distribution of the classes that are within the training set, a decision matrix is created. Based on the following matrix, each obtained node would be provided with a labeled class. Internal node testing is created using measurement for impurity. It is created for selecting threshold values and features. A known measurement of impurity for CART is the entropy impurity, and is mathematically represented below:

$$Entropy(s) = -\sum_{i=1}^{k} P(\frac{i}{s}) \log_2 P(\frac{i}{s}) \tag{1}$$

In the above equation, $Entropy(s)$ denotes the entropy value at s, $p(\frac{i}{s})$ is the relatively distribution of class i at s, and k denotes the number of classes. The optimal splitting value for (s) is selected from a splitting value group (y); thus, the highest impurity is the impurity difference between nodes of root and

Journal Pre-proof

children. The equation is as follows:

$$\triangle E(y,s) = E(s) - (P_L E(_{SL}) + (P_R E(_{SR})) \tag{2}$$

In the above equation, $\triangle\ E(y,t)$ denotes the the impurity drop, $E(_{SL})$ and $E(_{SR})$ are the nodes of the right and left branches impurities, and P_L and P_R are the probability of input to be in the right (SR) or left (SL) child nodes.

Table 2 describes the significance of features chosen by the CART approach, where features F34, F35, F36, F37, F26, F27, F6, F8, F31, F32, F33, F9, F15, F19, F20, F25, F1, F2, F3, F7, F6, and F43 can be used for distinguishing between malicious and legitimate connections, and features F13, F14, F16, F17, F18, F21, F4, F5, F10, F11, F12, F24, F22, F41, F23, F30, F29, F38, F39, F28, F40, and F42 cannot be used to distinguish between malicious and legitimate connections. The feature selection process is carried out based on the input samples' contributions, and the feature's significance is determined based on each input sample's role. For instance, it may be a surrogate or a primary splitter. Surrogate splitters serve as backup rules that simulate the main rules splitting process. As for the features that can be used to distinguish between malicious and legitimate connections, they will be utilized for generating host features during the feature extraction phase.

Table 2: Feature ranking

Feature	Significance	Feature	Significance	Feature	Significance
$\tilde{F1}$	0.812	$\tilde{F15}$	0.718	$\tilde{F32}$	0.551
$\tilde{F2}$	0.810	$\tilde{F19}$	0.703	$\tilde{F33}$	0.531
$\tilde{F3}$	0.787	$\tilde{F20}$	0.660	$\tilde{F34}$	0.509
$\tilde{F6}$	0.774	$\tilde{F25}$	0.619	$\tilde{F35}$	0.449
$\tilde{F7}$	0.763	$\tilde{F26}$	0.600	$\tilde{F36}$	0.371
$\tilde{F8}$	0.754	$\tilde{F27}$	0.573	$\tilde{F37}$	0.286
$\tilde{F9}$	0.743	$\tilde{F31}$	0.566	$\tilde{F43}$	0.194

Journal Pre-proof

3.2.3. Host feature extraction at network level

Table 3 shows the 16 host features collected using the proposed approach. The approach is based on the following three observations. First, bot infected hosts share particular malicious behavior, and the features differ from those of a normal host [58]. Second, the Bot's behavior during propagation repeats itself in a frequent manner since it is attempting to infect multiple hosts [10, 59]. Third, a software program generates the Bot connections [60].

For the feature extraction phase, it may start immediately in the event that the packets are transferred between the hosts. To extract the features of a node in a manner that is more accurate, we need to collect sufficient network traffic; otherwise, feature extraction is not going to be sufficiently robust. Thus, the hosts' behavior in the proposed approach is observed by analyzing their traffic packets during the sliding window time. This allows us to obtain the required number of packets. During the feature extraction phase, each network host has a distinctive feature record. After that, the host feature record can then be used to differentiate between legitimate network traffic and malicious botnet traffic. This can be achieved using online machine learning techniques, as well as reinforcement techniques.

Journal Pre-proof

Journal Pre-proof

Table 3: Host features of network traffic

Features	Description
F1	Total transmission flows per host in time-window.
F2	Total transmission unique connections per host in time-window.
F3	Total connections try per host in a time-window.
F4	High-severity of dest. port rates in time-window.
F5	The rate of use of unique dest. ports per host in time-window.
F6	The rate of use of unique source ports per host in time-window.
F7	The rate of transmission of unique host connections in time-window.
F8	High-severity of source port rates in time-window.
F9	Failures connections rates per host in a given time interval.
F10	Control packets Entropy rate for connections per host in time-window.
F11	Received Control packets entropy rate for connections per node in time-window.
F12	Transmitted Control packets entropy rates for connections per node in time-window.
F13	The avg. time between host connections.
F14	The avg. time between source inter-arrival control packets.
F15	The avg. length of the connection.
F16	Dispersion index.

Port scanning is the most common activity that precedes a cyber attack, as well as during various stages of a bot malware lifecycle (e.g. attack and propagation). For instance, during the propagation phase, a bot seeks to discover and contacting other peer bots within the same network. Thus, analyzing and monitoring the rate of the connections that are newly established can facilitate the detection and measurement of any malicious bot behavior. Computer ports are subdivided into two main classes, namely: low-severity and high-severity ports. Based on the information issued by the Dshield Organization [61], high-severity ports include those that are most likely to get scanned. All other ports are considered as ports of low-severity. Thus, the present study uses the port scanning method for detecting malicious cyber activities, and features F1 to F8 reflect the scanning behavior.

There are a number of botnet traffic behavior traits, such as bots showing a connection failure in the network. For example, when a bot connects to the botnet network, it must find a point of entry that may be a peer host or a C2C server, in order to deliver information about its current situation and receive

Journal Pre-proof

new instruction(s). Consequently, if any peer attempts to create a connection with those hosts, it may lead to a connection failure. The feature of connection failure (F9) that is based on the TCP connection shall be labeled as failed, in the event that the three-step handshake is not complete [62].

The number of control packets for legitimate network traffic is observed to have higher diversity in comparison to bot connection traffic. This is because users may use applications that have very different behavior for control packets. Thus, we do not expect to find trends in the frequency of the control packet. However, during the peer discovery stage, bots will attempt to contact other botnet peers, and hence that is a repeat onnection behavior. Such a behavior trait is telling, and we can use an entropy algorithm [63] to measure the randomness or amount of entropy that is within the control packet variation per host. The latter can then be used to model the control packets number connected to the node as a discrete symbol. High entropy implies a legitimate connection, and a low entropy may suggest a botnet connection. Therefore, further investigation is required. The entropy of the frequency of the control packet per host (F10 to F12) is estimated through a group $Cp = [c1, c2, \ldots .cn]$, where ci refers to the number of control packets per connection. This is mathematically expressed as follows:

$$E(t) = -\sum_{i}^{n} ci \log ci \tag{3}$$

Features F13 to F15 are related to the network host inter-arrival control packets. The time of inter-arrival packet refers to the time needed for creating and transferring data to the transport layer by the application [64]. This time is calculated by gathering the time from two consecutive packets in the same connection. The focus of the proposed technique is the host features, which are estimated at the network. The target of the proposed technique is represented in detecting an infected machine. The focus of the proposed approach is, therefore, on the time between the packets from the host.

For feature F16, we use the index of dispersion for counts (IDC) to measure

Journal Pre-proof

the probability distribution dispersion for the packets sent by the host. Gusella [65] emphasize on the significance of applying the latter index in the identification of packet variability. This index is used to quantify whether an observed group is dispersed or clustered with a standard statistic model. IDC refers to the variance (σ)-to-mean (μ) ratio, as expressed below:

$$IDC = \frac{\sigma^2}{\mu} \tag{4}$$

3.3. Malicious activity detection

Malicious activity detection includes an offline stage (training), an online detection stage, and a reinforcement-learning stage. In the first stage, the classifier is provided with a group of legitimate and bot feature vectors for the purpose of training. When the training ends, newly extracted features are uploaded in order to classify the hosts' activities within the network as legitimate or malicious.

A neural network is utilized to serve as a detector to identify malicious activity(ies), since the network has robust capabilities for non-linear system control and identification. That is attributed to an inherent capability of approximating an arbitrary nonlinear problem [66, 67, 68]. The resilient back-propagation-learning algorithm is used for neural network training, in order to reduce the negative impacts of the fractional derivatives' volume. The derivative is used merely for locating the weighted update's trend. As for the derivative's volume, it does not have any negative role in the weight updating process. The size of the weight change can be identified using the formula listed below [69]:

$$\triangle w_{ij}^{(t)} = \begin{cases} -\triangle_{ij}(t), & if \frac{\partial E(t)}{\partial w_{ij}} > 0 \\ -\triangle_{ij}(t), & if \frac{\partial E(t)}{\partial w_{ij}} < 0 \\ 0, & else \end{cases} \tag{5}$$

In the above equation, $\triangle_{ij}(t)$ refers to the change in weight between the hidden and input layers that are within the current iteration (t). $\frac{\partial E(t)}{\partial w_{ij}}$ refers to

Journal Pre-proof

the partial derivative of each weight. After having the weights calculated, the newly updated weight value shall be set. That is performed using the formula listed below:

$$\triangle_{ij}^{(t)} = \begin{cases} \eta^{+} \cdot \triangle_{ij}(t), & if \frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} > 0 \\ \eta^{-} \cdot \triangle_{ij}(t), & if \frac{\partial E(t-1)}{\partial w_{ij}} \cdot \frac{\partial E(t)}{\partial w_{ij}} < 0 \\ \triangle(t-1), & else \end{cases} \tag{6}$$

$\triangle_{ij}^{(t)}$ refers to the updated value of the current iteration t, and $\eta+$ refers to the positive step value (usually 1.2). As for $\eta-$, it refers to the negative step value (usually 0.5 [69]). The neural network classifier used in this study includes 16 inputs and 2 output parameters. In order to avoid over-fitting by employing several hidden layers, we use the technique described in [70] to decide on the number of hidden layers neurons.

From Figure 2, one can observe during the offline stage, a group of identified malicious and legitimate attribute vectors is added to the classifier. This is done for training our detector in order to classify the host behaviors on the network as either malicious or legitimate. In order to ensure that quality of the learned neural network agent, a cross–validation technique is used to estimate the classifiers' error rate. Through cross-validation, the dataset can be randomly partitioned into several N samples, where evaluations are run for N iterations. At each iteration, $N-1$ samples can be chosen to train the model. As for the last fold of samples, it shall be applied for evaluating the classifier' accuracy.

Journal Pre-proof

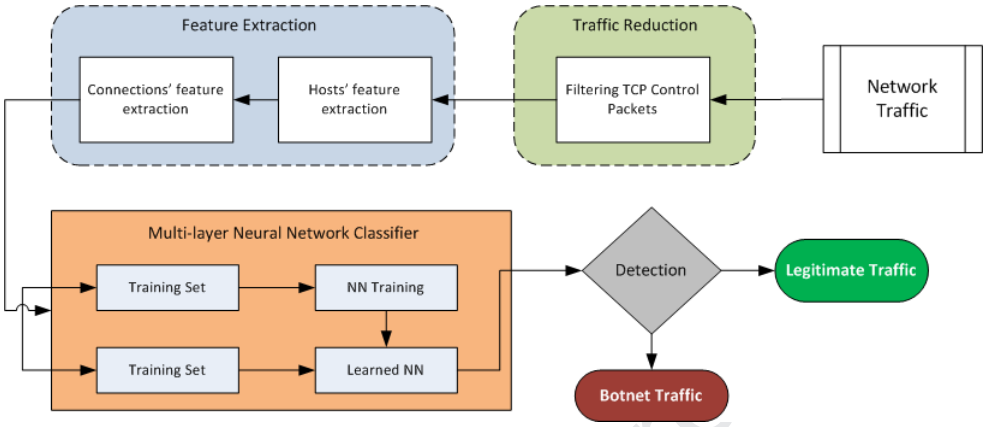

Figure 2: Off-line Phase

In the online detection stage, the agent (trained neural network) will continuously classify the host within the network. Then, the agent sends a report to the network administrator about the activities of the hosts. In addition, as observed in Figure 3, the reinforcement learning agent simultaneously operates to extract new features that shall participate in improving the performance level of the detection agent in the future.

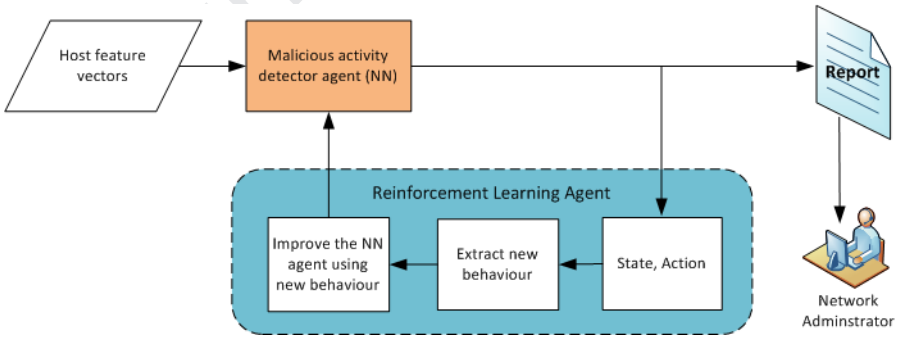

Figure 3: On-line Phase

3.4. Bot detection using reinforcement learning

Reinforcement learning (RL) techniques are widely used to handle problems that involve difficulty in determining the solution explicitly, provided that it is probable to generate the signals of reward [71, 72]. This applies in our botnet

Journal Pre-proof

detection problem. Specifically, the RL 'obstacle' is expressed in the partially observable markov decision process (POMDP) context. POMDPs are usually employed for representing dynamic systems, which include the systems used for the detection of botnet.

A POMDP is a group of states (S), which characterizes the status of the neural network agent (NN_{St}), controller agent (AG_{St}), and host (H_{St}) states. (NN_{At}) denotes the actions of agent at time t, and the agent of neural networks selects actions using π policy. NN^{π} (H_{St}, A) denotes the possibility of having the agent selecting action A when the host is in the state (H_{St}). $R(AG_{St})$ represent the estimation of reward function (T_{St}) denotes the transition function of the controller agent system.

The Markovian transition function defines the system's dynamics. It also generates the possibility $T(AG_{St}, NN_{At}, AG_{St+1})$ of transitioning into an agent state AG_{St+1} after action NN_{At} is taken in state AG_{St}. The reward function shall assign the new host's state (H_{St}) number, and the overall amount of the system's host states shall be processed as a numeric value to agent state (AG_{St}).

At any time, the POMDP can be representing the system's state. When the action is chosen by the neural network agent (NN_{At}), the controller agent rewards and the host state value shall be estimated. After that, based on the collected reward's size, the controller agent's transition function (T_{St}) changes the neural network agent, and thus resulting in a new state NN_{St+1}. In our context, the detection of a P2P bot is expressed as an RL problem. That includes a selection for the action space, reward, value state, and transition functions.

In the action space, after having the action space defined, the host on the network at each time-window is provided with the possibility of being a bot or legitimate node. Then, the RL agent shall take that possibility into account. This is also done for estimating the state's reward.

In the agent reward function, at any time step, the reward signal is equivalent to the quantity of the new states that are processed by the hostswithin the network via the time-window. The signal of reward calculates the importance

Journal Pre-proof

of the new state, by utilizing the value of state function in several time-windows. In our context, the new state may be a bot or a legitimate node.

In the value state function, all the (H) nodes within the network have several states based on the use mode. As for the function of the value state, it represents the prospect reward of each host (H_{St}), according to the policy NN^{π}. In every time-window, the output of the neural network for every host state is split into two sub-states of probabilities. These sub-states are legitimate $E(L)$ and malicious $E(M)$. Thus, the output of each host state is expressed as $(E.M(H_{St}))$ or $(E.L(H_{St}))$, and the formula below identifies the value state function evaluation for legitimate and bot hosts and controller agent.

- Assessment the value of state function for bot hosts:

$$EV^{\pi}(H(M)) = \frac{\sum_{i=0}^{n} E(M_{st(i)})}{n} \tag{7}$$

  $EV^{\pi}(H(M))$ denotes the percent of bot activity expected for the host in n time-windows, and $E(Mst(i))$ is the possibility of malicious activity results using the policy of the existing neural network.

- Assessment the value of state function for legitimate hosts:

$$EV^{\pi}(H(L)) = \frac{\sum_{i=0}^{n} E(N_{st(i)})}{n} \tag{8}$$

  $EV^{\pi}(H(L))$ denotes the expected percent of host's legitimate activity in n time-windows, and $E(L_{st(i)})$ is the possibility of legitimate activity results based on the policy of the existing neural network.

- Assessment of controller state value agent function:

$$V(s) = V(s) + \begin{cases} V(M_{st}) = argmax(M(Actions)) & EV^{\pi}(H(M)_{st}) > EV^{\pi}(H(L)_{st}) \\ V(L_{st}) = argmax(L(Actions)) & EV^{\pi}(H(M)_{st}) < EV^{\pi}(H(L)_{st}) \end{cases} \tag{9}$$

Journal Pre-proof

In the above equation, $V(s)$ denotes the calculated states which obtain the highest reward using the existing neural network agent policy NN^{π}.

Finally, for the function of transition, all techniques in the RL domain require the implementation of a policy that guarantees that a balance is achieved between exploitation and exploration. The problem lies in identifying the way to find an effective policy for action-selection. This policy should be based on sufficient exploitation and exploration data.

In our context, we attempt to find an effective method in order to strike a balance between exploitation and exploration. Thus, a directed exploration approach is adopted, which allows us to explore the state and action as much as possible before shifting to another approach.

The most straightforward directed exploration method is a greedy technique. In any host state, the state that shows the highest exposure of probability value shall be selected. In addition, after implementing the explorative strategy, several steps are followed, in order to identify the hidden goals. In case the target is a novel state to the system, the system can simply shift from exploration to exploitation.

The function of the transition is expressed below:

$$T_{st} = \frac{\sum newV(s)}{\sum V(s)} >= \theta \tag{10}$$

In the above equation, T_{St} denotes the rate of exploring $newstateV(s)$ to the whole of the state $V(s)$. Therefore, T_{St} depends on the analyzed network traffic volume. The adaptable threshold (θ) is set by the network admin. It is set based on the desired degree of network protection. For example, in sensitive networks, θ is very short. A low θ value also indicates that there is a high learning rate. Table 4 summarizes the RL system parameter and symbols.

Journal Pre-proof

Table 4: RL parameters and symbols

RL symbol	Description
S	Current state of environment.
A	Agent Action.
π	Action policy.
θ	Threshold transition value.
H_{St}	Host state at time (t).
NN_{St}	State of the neural network agent at time (t).
NN_{At}	Actions of agent at time (t).
NN^{π}	The policy of neural network.
AG_{St}	Controller agent state at time (t).
$R(AG_{St})$	Represent the estimation of reward function at current state and time (t).
T_{St}	Represents the transition function of the controller agent system.
$V(s)$	The value of state s, using the current neural network agent policy NN^{π}.
$NN^{\pi}(HSt; A)$	Represents the possibility for the agent to selected action A when the host is in the state (HSt).
$EV^{\pi}(H(L))$	Represents the probability rate of a legitimate behavior using the current neural network agent policy.
$EV^{\pi}(H(M))$	Represents the probability rate of a malicious behavior using the current neural network agent policy.

Algorithm 1 describes our proposed system. First, it extracts a new behavior from the environment. After that, it decides on the action to take, on the basis of the present policy of the neural network. As for vector V, it is employed to gather observation for each new states and actions. Whenever the agent collects a sufficient number of new states, it moves to the state of exploitation. This is done in order to utilize those states. At last, the main control agent assesses the new neural network agent's performance.

Journal Pre-proof

Journal Pre-proof

Algorithm 1: Bot detection using RL technique

Input: Host state (neural network outcomes).

Output: New updated neural network

$V_{(s)} = 0.$; $T_{st} = 0.$;All_Dataset = RefDataset.;Temp_Dataset = 0.

Read the environment observation($state(S_t)$).

Perfrom action $NN^{\pi}(A|(S_t, S_{t+1}))$.

Extract the reward (R).

Estimate the probability of a Bot node:
$EV^{\pi}(H(M)) = \frac{\sum_{i=0}^{n} E(M_{st(i)})}{n}$.

Estimate the probability of a legitimate node:
$EV^{\pi}(H(L)) = 1 - EV^{\pi}(H(M))$.

Extract the state with highest expected reward:

$$V(s) = V(s) + \begin{cases} V(M_{st}) = argmax(M(Actions)) & EV^{\pi}(H(M)_{st}) > EV^{\pi}(H(L)_{st}) \\ V(L_{st}) = argmax(L(Actions)) & EV^{\pi}(H(M)_{st}) < EV^{\pi}(H(L)_{st}) \end{cases}$$

Estimate the amount of expected reward:
$T_{st} = \frac{\sum newV(s)}{\sum V(s)}$

if *($T_{st} \geq \theta$)* **then**

  Temp_Dataset = Temp_Dataset + $V(s)$.

  Reset $V(s)$.

  ($NN^{\pi}2$): Creation and evaluation:

  - Create a new Neural Network $NN^{\pi}2$ using Temp_Dataset.
  - Evaluate the performance of $NN^{\pi}2$ using cross-validation techniques.
  - Evaluate the performance of $NN^{\pi}2$.

  if *($NN^{\pi}2$ pass the validation test)* **then**

    - $NN^{\pi} = NN^{\pi}2$.
    - All_Dataset = All_Dataset + Temp_Dataset.
    - Reset Temp_Dataset.
    - **goto** to Step 2

  else
    goto Step 1.

else

  goto Step 2.

Journal Pre-proof

The proposed approach's key advantage is that the approach sticks to a specific strategy for a certain period. It will not end up taking 1-step in the direction of exploratory nor 1-step in the other direction. Management of the rate of learning (exploring) a new behavior (state) depends on the network traffic's state. In comparison with low network traffic, if there is a significant volume of network traffic, the controller agent gains numerous new states. After setting the most useful amount of reward by the system, the system shifts to the exploitative strategy. This is done by creating a new dataset, by combining new extracted states with the old dataset. The new dataset is then used for training. First, a cross-valuation technique is adopted for assessing the performance of a new neural network (NN) agent. It is also adopted for assessing the performance evaluation matrices, namely: Matthews correlation coefficient (MCC), accuracy(ACC), area under the ROC(AUC), and root means square error (RMSE). Second, the new NN agent is assessed usingthe old reference dataset (action and state), in terms of AUC, MCC, ACC and RMSE. Third, in case the system has the assessment test, the system's primary controller will replace the detection agent with a new one (NN agent). However, when the new NN agent has a low achievement level, the system shall retain the current NN agent. It shall also reset the action and new state buffer.

In summary, the system contains three NN agents. In the first NN, the reference dataset is utilized to train the first initial agents. The second neural network is established through the use of newly extracted features (states) from the environment. Finally, the best configuration of the neural network that passes the assessment process is utilized in the detection phase.

4. Experimental evaluation and results

4.1. Datasets and Tools

We use three primary datasets to evaluate the proposed system, which include non-malicious and malicious traffic – see Table 5. The first dataset is the information security and object technology (ISOT) dataset [73], which includes

Journal Pre-proof

Storm Bot, Waledac Bot, and non-malicious traffic. The second dataset includes four legitimate for P2P applications (i.e. Vuze, uTorrent, Frostwire, and eMule), and the traffic of three P2P Botnets (i.e. Zeus, Storm and Waledac) [32]. The third dataset is in the Information Security Centre of Excellence (ISCX) dataset [74], which contains legitimate network traffic.

Table 5: Datasets

Group	Traffic source	Purpose	Duration	packets number
G1	Strom Bot [73]	Training	3115 seconds	128241
G2	Waledac Bot [73]	Training	605 seconds	118379
G3	Normal traffic [73]	Training	126273 seconds	564999
G4	eMule - [32]	Training/Testing	24 hours	6736353
G5	uTorrent - [32]	Training/Testing	24 hours	6278385
G6	Vuze - [32]	Training/Testing	24 hours	11732688
G7	FrostWire - [32]	Training/Testing	24 hours	4429535
G8	Normal traffic – [74]	Testing	24 hours	3776931
G9	Strom Bot traffic - [32]	Testing/(Zero-Day attacks)	24 hours	4251875
G10	Waledac Bot traffic - [32]	Testing/(Zero-Day attacks)	24 hours	12915757
G11	Zeus Bot traffic - [32]	Testing/(Zero-Day attacks)	24 hours	114548

The experiments are carried on an Intel Xeonprocessor with a six-core monster clocked at 2.1GHz (with a 2.6GHz Turbo) and 64 GB RAM, and the proposed approach is implemented using Matlab 2018b. Table 6 summarizes the tools used in the experiments.

Journal Pre-proof

Table 6: Experimental Tools.

Name	Description	Version
Jpcap [49].	Java library for capturing and sending network packets.	0.7
Tcpreplay [75].	Replays Pcap files onto the network	3.4.4

4.2. Setup

An experimental dataset is created for evaluating the approach's capability in the online detection of a new bot infection. In order to simulate a realistic network traffic condition, a testbed is constructed for replaying malicious botnet traffic, P2P application traffic, and normal daily activity traffic, using TcpReplay. Then, the JPCAP tool is used to capture the replayed network traffic.

The setup comprises the following five steps:

(1) Replaying the entire legitimate and malicious trace files, and capturing packets through the use of different time-window sizes.
(2) Reducing network traffic through the use of the proposed network traffic reduction method.
(3) Extracting vectors of feature for hosts while capturing of packets.
(4) Obtaining the classification results through the use of the testing sets and the prepared training, by adopting the proposed technique.
(5) Identifying the size of time-window that provides higher levels of detection performance and stability during the online and offline stages.

Due to the significant volume of network packets to be analyzed, network traffic is divided into time-windows. In addition, time-window is required for delivering the results to the admin of network on a timely basis. The use of time-window shorter than 10 seconds is avoided due to having few captured packets that are incapable of showing the characteristics of the traffic behavior. We also avoid using time window greater than 60 seconds, so that we can detect the bot

Journal Pre-proof

as early as possible. Bots can generate a temporal behavior after the infection phase [50]. Thus, such a behavior is leveraged to acquire the requisite bot behaviors in the time-window. Hence, in this paper, we begin with a ten seconds time-window, which is incremented gradually. This allows us to determine an optimal performance level. Additionally, 10% of the time-window size is used to slide between time-windows to quickly detect any malicious activity, rather than idling for the entire next window.

In order to assess the proposed system's performance level, the following metrics need to be computed:

- True positive (TP): The number of bot samples labeled as malicious.
- True negative (TN): The number of normal samples labeled as legitimate.
- False positive (FP): The number of normal samples labeled as malicious.
- False negative (FN): The number of bot samples labeled as legitimate.

The false positive rate (FPR) denotes the rate of legitimate samples that are misclassified as botnet samples, and is mathematically expressed as follows:

$$FPR = \frac{\sum FP}{TN + FP} \tag{11}$$

The detection rate (DR) is mathematically expressed as follows:

$$DR = \frac{TP}{TP + FN} \tag{12}$$

Accuracy (ACC) is the rate where samples are correctly classified, and is mathematically expressed as follows:

$$ACC = \frac{TP + TN}{TN + TP + FN + FP} \tag{13}$$

Precision is the rate where bot samples correctly classified, and is mathematically expressed as follows:

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

Journal Pre-proof

The $F-measure$ is used to measure the accuracy level of the test, and both recall and precision of the test are taken into consideration when calculating the score, as shown below:

$$F-measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{15}$$

The root mean square error ($RMSE$) is the difference between the actual value estimated by the method of detection and the target value, and is mathematically expressed as follows:

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(yi - ti)^2}{N}} \tag{16}$$

In the above equation, N denotes the number of input samples, and yi denotes the model's actual output. $RMSE = 0$ indicates that the model's output matches the targets.

The non-dimensional error index $NDEI$ is applied to evaluate the quality of prediction, and is mathematically expressed as follows [76]:

$$NDEI = \frac{RMSE}{Std(ti)} \tag{17}$$

The Matthews correlation coefficient (MCC) is adopted to estimate classifier efficiency in the event of an imbalanced dataset [77], and is mathematically expressed as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{18}$$

The receiver operating characteristic (ROC) is a graphical representation, which shows the binary classifier efficiency. The x-axis represents the FPR, and the y-axis represents the TPR. The area under the ROC (AUC) denotes the performance of the classifier [78]. In addition, the AUC is generally considered a much more robust estimator of the classifier's performance level [79].

Journal Pre-proof

4.3. Network traffic reduction evaluation

Our proposed network traffic reduction algorithm is designed to minimize the volume of network traffic that needs to be examined in our system. In flow-based detection schemes, such as those of [32, 41, 80, 19], every packet is analyzed. However, we argue that this is not realistic to be implemented in real-time high-speed network. As shown in Table 7, the network traffic reduction algorithm can decrease the normal traffic by an average of over 50%.

Table 7: Packets reduction rates

Group	# control packets	traffic reduction rate
G1	64551	0.5033
G2	69936	0.5907
G3	226308	0.4005
G4	2780725	0.4127
G5	4237135	0.6748
G6	741677	0.6321
G7	2406066	0.5431
G8	1686962	0.4466
G9	1169900	0.2751
G10	9395310	0.7274
G11	59255	0.5172

The rates of control packets derived from each rule of traffic reduction technique are presented in Table 8. We also remarked that several detection systems proposed in the literature have good detection rates [81, 82, 18, 20, 21]. However, these approaches are not designed for large networks, partly due to their reliance on DPI techniques [83]. For instance, BotHunter [81] employs a signature-based detection engine and payload-based anomaly detector. Rishi [82] and BotSniffer [18] require the parsing of IRC communication content. TAMD [21] inspects the payloads of packets, with the aim of estimating similarities between content.

Journal Pre-proof

Our proposed network traffic reduction approach, however, focuses only on a small portion of the TCP packets that are utilized in connection initialization.

Table 8: Network packet reduction rates based on rules

Group	R(1)	R(2)	R(3)	R(4)	R(5)	R(6)
G1	34.70%	7.00%	13.00%	8.70%	16.70%	20.00%
G2	30.00%	11.20%	8.80%	6.70%	21.30%	22.00%
G3	22.50%	25.80%	20.50%	17.80%	8.30%	5.00%
G4	20.0%	26.70%	15.80%	17.50%	13.30%	6.70%
G5	26.70%	22.20%	22.80%	10.70%	7.70%	10.00%
G6	25.00%	21.30%	18.30%	18.20%	5.20%	12.00%
G7	26.70%	19.30%	24.00%	15.00%	8.50%	6.50%
G8	23.80%	20.00%	19.0%	22.70%	6.70%	7.80%
G9	29.80%	8.20%	8.50%	8.80%	22.80%	21.80%
G10	29.00%	15.00%	6.00%	8.00%	17.00%	25.00%
G11	30.20%	7.70%	4.00%	9.80%	20.50%	27.80%

Table 9 presents a comparative summary for the performance of our detection approach with other competing approaches [32, 41, 80, 19, 84].

Table 9: Comparison of network traffic reduction rates

Method	Traffic Reduction	TPR	FPR
PeerRush [32]	None	99.10%	0.10%
Zhao et al. [41]	None	98.10%	2.10%
Timothy et al. [80]	None	92.0%	11.0-15.0%
Gu et al. [19]	None	1.0%	0-6%
Wang et al. [84]	>70.0%	95.0%	0-3.0%
Proposed approach	(40.0-70.0)%	99.10%	0.010%

Journal Pre-proof

4.4. Host feature evaluations

Table 3 presents the host features of network traffic, and Min-Max normalization [85] is used to calculate the normalized average value of each feature.

$$Y' = \frac{Yi - Ymin}{Ymax - Ymin} \tag{19}$$

In the above equation, Y' is the normalized value of Yi, $Ymin$ is the vector of the minimum feature value, and $Ymax$ refers to the vector of the maximum feature value.

Figure 4 presents the average normalization value distribution for each feature. We observe that the distribution of normal host traffic and bot host traffic. For example, as illustrated in Figure 4 and explained in Section 3.2.3, features F12, F15, F5, F10 and F16 are considered to be discriminate features that facilitate botnet detection.

Pre-proof

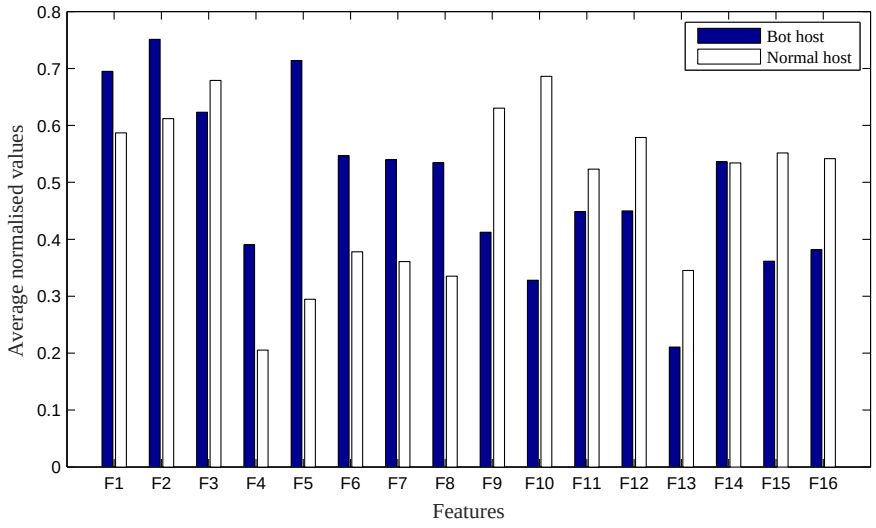

Figure 4: Normalized feature comparison

Figure 5 presents a variation between bot and legitimate network flows, based on the total number of control packets' entropy rates per host. The figure indicates that the entropy rates for the legitimate host are within the range of

Journal Pre-proof

0 to 5, while these rates are below 0.5 for a bot host traffic. The difference in entropy rates between a bot and legitimate hosts is attributed to the presence of the bot due to the regularity in the count of control packets per flows. As for the legitimate host flows, it shows that there is diversity and irregularity in the control packet count per flows. As a result, the legitimate hosts show an entropy value that is high, whereas the bot shows an entropy value that is low.

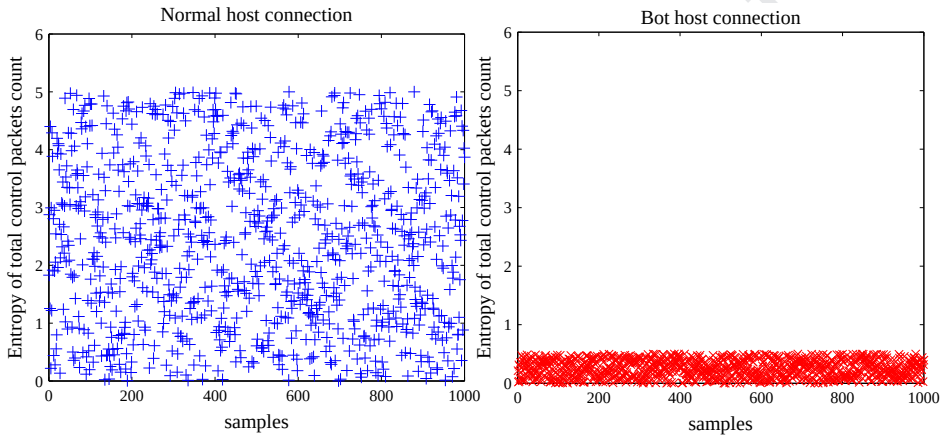

Figure 5: Control-packets Entropy rates

4.5. Evaluation of offline phase

Figures 6 and 7 present the result of assessing the trained agent during the offline phase of the proposed approach, where the x-axis is the size of time-window applied for phases of the feature extraction. Based on the 60 seconds time–window, the proposed approach outperforms the other approaches, in terms of detection, accuracy and F-measure rates (i.e. 99.0%, 98.30% and 98.90%, respectively). Furthermore, the lowest FPR has a 60 seconds time-window size. In the meantime, a 10 seconds time frame shows the lowest performance – see Figure 6.

Journal Pre-proof

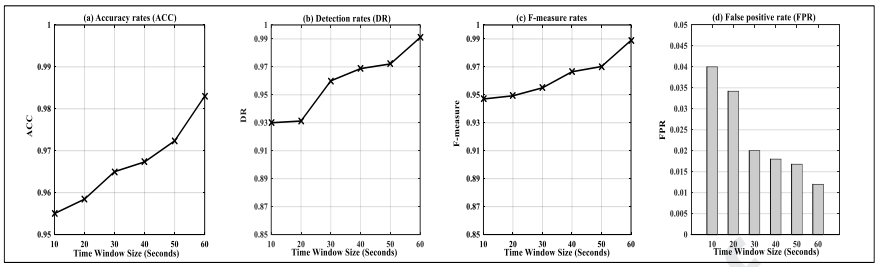

Figure 6: Offline phase evaluation(ACC,DR,F-measure and FPR)

Figure 7 presents the AUC, MCC, RMSE and NDEI of the bot detection system, based on different time-window sizes. We observe thatMCC and AUC rates are the highest rates, respectively at 95.60% and 99.10% based on the 60 seconds time-window.

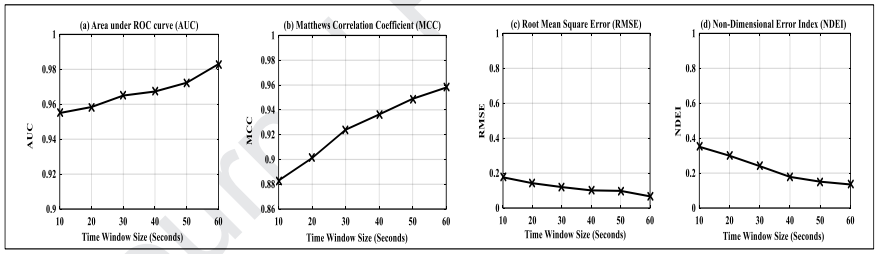

Figure 7: Offline phase evaluation (AUC, MCC, RMSE and NDEI)

We also observe that the 60 seconds time-window has good performance with good result stability, due to the small time-window size. In addition, bots generate a temporal behavior after the phase of infection [50]. Thus, 60 seconds are appropriate for capturing the network traffic that can be used to facilitate accurate classification. As shown in Figures 6 and 7, the proposed approach can detect P2P bots with a high accuracy rate (associated with a low FPR). We emphasized that these results are obtained using only the training dataset, and the focus of the offline stage is to prepare the classification agent for online work.

Journal Pre-proof

4.6. Evaluation of Online Bot Host Detection Approach

We will now describe the findings of the proposed approach on the test dataset (Zero-day attack) – see Figures 8 and 9 present the overall results gained from the online experimental result analysis. As observed, the proposed approach uses an online evaluation and has the highest F-measure, accuracy, and detection rates respectively at 98.80%, 98.30% and 97.90%, based on a time-window size of 60 seconds. The 10 seconds time-window size yields the least desirable performance.

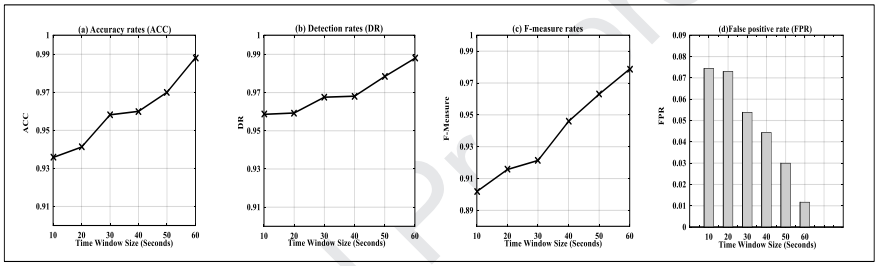

Figure 8: Online phase evaluation (ACC,DR,F-measure and FPR)

As observed from Figure 9, the MCC and AUC rates are 97.6% and 99.96% respectively. Thus, these experimental results show that the performance of our proposed online detection system is capable of handling imbalanced dataset in the 60 seconds time-window. Furthermore, RMSE and NDEI are also used evaluated, and clearly at the 60 seconds time-window size, both RMSE and NDEI achieve 0.093 and 0.187% respectively.

Journal Pre-proof

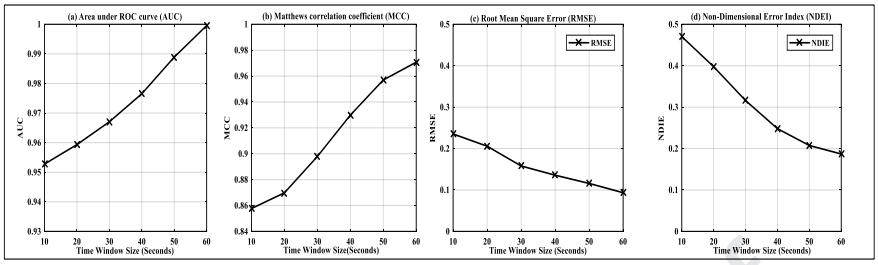

Figure 9: Online phase evaluation (AUC, MCC, RMSE and NDEI)

It is clear from Figures 9 and 9 that the proposed method has a good performance result at the time-window size of 60 seconds. Also, to evaluate the performance of our proposed approach, we examine the ROC curve – see Figure 10. Again, the 60 seconds time-window size has the highest rates for the classification of legitimate and bot traffic respectively at 0.991 and 0.98.

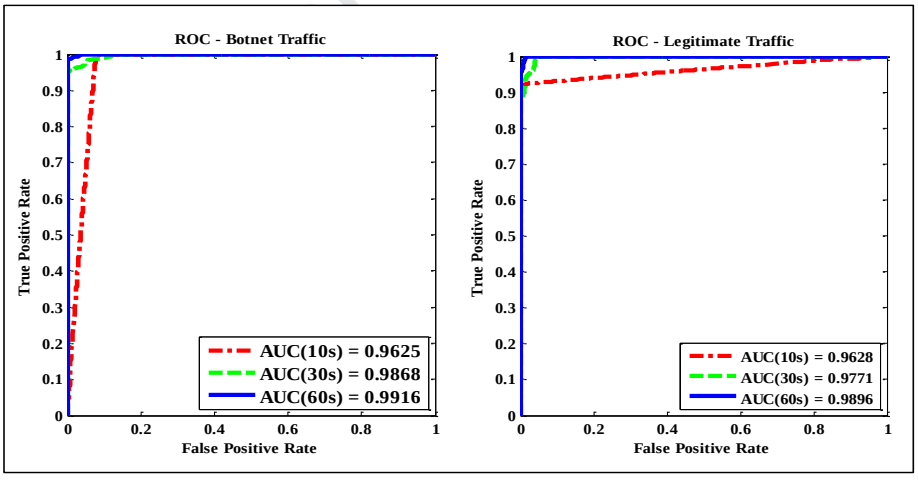

Figure 10: ROC comparison

4.7. New botnet detection

To assess the performance of the proposed approach in detecting novel types of P2P bots, a sample was selected. The sample consists of Storm, Zeus, and

Journal Pre-proof

Waledac. As shown in Figure 11, the proposed approach is effective in detecting new P2P bots. In Figure 11(A), for example, the rates of detecting Zeus Bot is lower than those of Waledac and Storm Bots at the 60 seconds time-window size (i.e. 93.8%, 98.2% and 96.83%, respectively). This is because we use both Storm and Waledac for the testing and training dataset.

As observed in Figure 11 (B), our proposed approach has low FPRs for Zeus, Storm and Waledac of 0.04%, 0.07% and 0.09% respectively, at the 60 seconds time-window size.

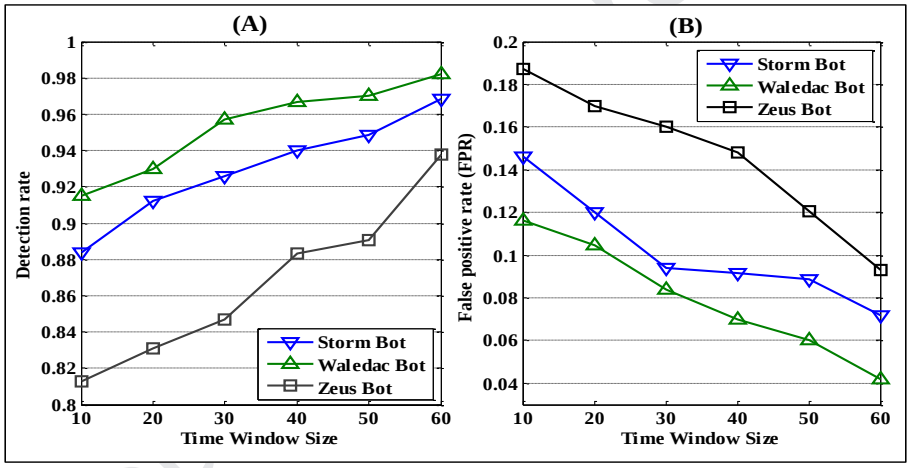

Figure 11: New botnet detection

A comparative summary of evaluating the proposed approach's performance with two other P2P botnet detection approaches [32, 41] is presented in Table 10. We use the dataset used by Zhao et al. [41] during the offline state, and the dataset used by Babak et al. [32] during the online state. We observe that the FPR and bot detection rates using our proposed approach are considerably better than those of the competing approaches.

Table 10: Performance evaluation: A comparative summary

Journal Pre-proof

Methods	FPR	DR	Network packet reduction rates
PeerRush [32]	0.10%	99.50%	0.0%
D Zhao et al. [41]	2.10%	98.10%	0.0%
Proposed online method	0.012%	98.30%	(40-70)%
Proposed offline method	0.01%	99.10%	(40-70)%

5. Conclusion and future work

Botnets remain an ongoing threat in today's networked society, and as bot malware evolves so do the mitigation strategies. Our proposed approach uses both reinforcement learning and our traffic reduction method. One key contribution of the proposed approach is the network traffic reduction technique, since we are able to reduce the input packets by about 50%. We demonstrate in the preceding section that our proposed approach has a detection rate of 98.30% and a low FPR of 0.010% at the 60 seconds time-window size. The bottleneck of bot detection using neural network is associated with the size and dimensionality of the dataset, as the number of the packets that require scanning is significant. This is where our proposed network traffic reduction approach plays a key role. The use of such an approach can reduce the training time required, and also increase the learning rate of newly extracted features in the online system. In addition, the proposed bot detection approach is shown to achieve good accuracy rate and is able to detect new bots.

However, there remains a number of challenges that need to be addressed. For example, bot masters will continue to explore ways of circumventing detection by existing approaches, for example using rootkits. In addition, botnets change dynamically through updates, and hence their operations may change after several life cycle stages. These characteristics are also known as the drifting concept [86]. Hence, the proposed approach adopts the idea of reinforcement

Journal Pre-proof

learning for dynamically improving the system throughout time. However, this is a rat race between future bot malware designers and botnet detection solution designers. Hence, there is a need to continue this line of research.

References

[1] S. Yu, Y. Tian, S. Guo, D. O. Wu, Can we beat ddos attacks in clouds?, IEEE Transactions on Parallel and Distributed Systems 25 (9) (2014) 2245–2254. doi:10.1109/TPDS.2013.181.

[2] S. Yu, W. Zhou, W. Jia, S. Guo, Y. Xiang, F. Tang, Discriminating ddos attacks from flash crowds using flow correlation coefficient, IEEE Transactions on Parallel and Distributed Systems 23 (6) (2012) 1073–1080. doi:10.1109/TPDS.2011.262.

[3] M. Alkasassbeh, M. Almseidin, Machine learning methods for network intrusion detection, CoRR abs/1809.02610.

[4] M. Almseidin, M. Alzubi, S. Kovacs, M. Alkasassbeh, Evaluation of machine learning algorithms for intrusion detection system, in: 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), 2017, pp. 000277–000282. doi:10.1109/SISY.2017.8080566.

[5] S. S. Silva, R. M. Silva, R. C. Pinto, R. M. Salles, Botnets: A survey, Computer Networks 57 (2) (2013) 378–403.

[6] W. Lu, G. Rammidi, A. A. Ghorbani, Clustering botnet communication traffic based on n-gram feature selection, Computer Communications 34 (3) (2011) 502–514. doi:http://dx.doi.org/10.1016/j.comcom.2010.04.007.
URL http://www.sciencedirect.com/science/article/pii/S0140366410001751

[7] A. Castiglione, R. D. Prisco, A. D. Santis, U. Fiore, F. Palmieri, A botnet-based command and control approach relying on swarm intelli-

Journal Pre-proof

gence, Journal of Network and Computer Applications 38 (2014) 22 – 33. doi:https://doi.org/10.1016/j.jnca.2013.05.002.
URL http://www.sciencedirect.com/science/article/pii/S1084804513001161

[8] K.-S. Han, E. Im, A Survey on P2P Botnet Detection, Vol. 120 of Lecture Notes in Electrical Engineering, Springer Netherlands, 2012, book section 56, pp. 589–593. doi:10.1007/978-94-007-2911-7_56.
URL http://dx.doi.org/10.1007/978-94-007-2911-7_56

[9] C. Ludl, S. McAllister, E. Kirda, C. Kruegel, On the Effectiveness of Techniques to Detect Phishing Sites, Vol. 4579 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, book section 2, pp. 20–39. doi:doi:10.1007/978-3-540-73614-1_2.
URL http://dx.doi.org/10.1007/978-3-540-73614-1_2

[10] J. Felix, C. Joseph, A. Ghorbani, Group Behavior Metrics for P2P Botnet Detection, Vol. 7618 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, book section 9, pp. 93–104. doi:10.1007/978-3-642-34129-8_9.
URL http://dx.doi.org/10.1007/978-3-642-34129-8_9

[11] C. Davis, J. Fernandez, S. Neville, Optimising sybil attacks against p2p-based botnets, in: Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on, 2009, pp. 78–87. doi:10.1109/MALWARE.2009.5403016.

[12] R. Weaver, Passive and Active Measurement: 11th International Conference, PAM 2010, Zurich, Switzerland, April 7-9, 2010. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, Ch. A Probabilistic Population Study of the Conficker-C Botnet, pp. 181–190. doi:10.1007/978-3-642-12334-4_19.

[13] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, L. Wang, On the analysis of the zeus botnet crimeware toolkit, in:

Journal Pre-proof

Eighth Annual International Conference on Privacy Security and Trust (PST), 2010, pp. 31–38. doi:10.1109/PST.2010.5593240.

[14] T. Holz, M. Steiner, F. Dahl, E. Biersack, F. Freiling, Measurements and mitigation of peer-to-peer-based botnets: A case study on storm worm, in: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats, LEET'08, USENIX Association, Berkeley, CA, USA, 2008, pp. 9:1–9:9.
URL http://dl.acm.org/citation.cfm?id=1387709.1387718

[15] C. Li, W. Jiang, X. Zou, Botnet: Survey and case study, in: Fourth International Conference on Innovative Computing, Information and Control (ICICIC) Fourth International Conference on, 2009, pp. 1184–1187. doi:10.1109/ICICIC.2009.127.

[16] A. Almomani, B. B. Gupta, S. Atawneh, A. Meulenberg, E. Almomani, A survey of phishing email filtering techniques, IEEE Communications Surveys Tutorials 15 (4) (2013) 2070–2090. doi:10.1109/SURV.2013.030713.00020.

[17] N. Moustafa, J. Hu, J. Slay, A holistic review of network anomaly detection systems: A comprehensive survey, Journal of Network and Computer Applications 128 (2019) 33 – 55. doi:https://doi.org/10.1016/j.jnca.2018.12.006.
URL http://www.sciencedirect.com/science/article/pii/S1084804518303886

[18] G. Gu, J. Zhang, W. Lee, Botsniffer: Detecting botnet command and control channels in network traffic (2008).

[19] G. Gu, R. Perdisci, J. Zhang, W. Lee, et al., Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection., in: USENIX Security Symposium, Vol. 5, 2008, pp. 139–154.

Journal Pre-proof

[20] J. Goebel, T. Holz, Rishi: identify bot contaminated hosts by irc nickname evaluation (2007).

[21] T.-F. Yen, M. Reiter, Traffic aggregation for malware detection (2008/01/01 2008). doi:10.1007/978-3-540-70542-0_11.

[22] O. Akinrolabu, I. Agrafiotis, A. Erola, The challenge of detecting sophisticated attacks: Insights from soc analysts, in: Proceedings of the 13th International Conference on Availability, Reliability and Security, ACM, 2018, p. 55.

[23] H. H. Pajouh, A. Dehghantanha, R. Khayami, K.-K. R. Choo, Intelligent os x malware threat detection with code inspection, Journal of Computer Virology and Hacking Techniques 14 (3) (2018) 213–223.

[24] H. Tran, C. Dang, H. Nguyen, P. Vo, T. Vu, Multi-confirmations and dns graph mining for malicious domain detection, in: Intelligent Computing-Proceedings of the Computing Conference, Springer, 2019, pp. 639–653.

[25] C. Wu, S. Sheng, X. Dong, Research on visualization systems for ddos attack detection, in: 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2018, pp. 2986–2991.

[26] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, A. Hotho, A survey of network-based intrusion detection data sets, Computers & Security.

[27] M. Singh, M. Singh, S. Kaur, Issues and challenges in dns based botnet detection: A survey, Computers & Security.

[28] Y. Zhauniarovich, I. Khalil, T. Yu, M. Dacier, A survey on malicious domains detection through dns data analysis, ACM Computing Surveys (CSUR) 51 (4) (2018) 67.

[29] M. Feily, A. Shahrestani, S. Ramadass, A survey of botnet and botnet detection, in: Third International Conference on Emerging Security Information, Systems and Technologies., 2009, pp. 268–273. doi:10.1109/SECURWARE.2009.48.

Journal Pre-proof

[30] H. Zeidanloo, M. Shooshtari, P. Amoli, M. Safari, M. Zamani, A taxonomy of botnet detection techniques, in: 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Vol. 2, 2010, pp. 158–162. doi:10.1109/ICCSIT.2010.5563555.

[31] W. Wu, J. Alvarez, C. Liu, H.-M. Sun, Bot detection using unsupervised machine learning, Microsystem Technologies 24 (1) (2018) 209–217. doi:10.1007/s00542-016-3237-0.
URL https://doi.org/10.1007/s00542-016-3237-0

[32] R. Babak, P. Roberto, L. Andrea, L. Kang, Peerrush: Mining for unwanted p2p traffic, Journal of Information Security and Applications 19 (3) (2014) 194–208. doi:http://dx.doi.org/10.1016/j.jisa.2014.03.002.
URL http://www.sciencedirect.com/science/article/pii/S2214212614000143

[33] D. M. J. Tax, One-class classification, TU Delft, Delft University of Technology, 2001.

[34] S. Garg, A. Singh, A. Sarje, S. Peddoju, Behaviour analysis of machine learning algorithms for detecting p2p botnets, in: 15th International Conference on Advanced Computing Technologies (ICACT), 2013, pp. 1–4. doi:10.1109/ICACT.2013.6710523.

[35] H. Jiang, X. Shao, Detecting p2p botnets by discovering flow dependency in c&c traffic, Peer-to-Peer Networking and Applications (2012) 1–12doi:10.1007/s12083-012-0150-x.
URL http://dx.doi.org/10.1007/s12083-012-0150-x

[36] H. Li, G. Hu, Y. Yang, Research on P2P Botnet Network Behaviors and Modeling, Vol. 307 of Communications in Computer and Information Science, Springer Berlin Heidelberg, 2012, book section 12, pp. 82–89. doi:doi:10.1007/978-3-642-34038-3_12.
URL http://dx.doi.org/10.1007/978-3-642-34038-3_12

Journal Pre-proof

[37] J. Zhang, R. Perdisci, W. Lee, U. Sarfraz, X. Luo, Detecting stealthy p2p botnets using statistical traffic fingerprints, in: IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN), 2011, pp. 121–132. doi:10.1109/DSN.2011.5958212.

[38] T. Zhang, R. Ramakrishnan, M. Livny, Birch: A new data clustering algorithm and its applications, Data Mining and Knowledge Discovery 1 (2) (1997) 141–182. doi:10.1023/a:1009783824328.
URL http://dx.doi.org/10.1023/A%3A1009783824328

[39] A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering: A review, ACM Comput. Surv. 31 (3) (1999) 264–323. doi:10.1145/331499.331504.
URL http://doi.acm.org/10.1145/331499.331504

[40] W.-H. Liao, C.-C. Chang, Peer to peer botnet detection using data mining scheme, in: International Conference on Internet Technology and Applications, 2010, pp. 1–4. doi:10.1109/ITAPP.2010.5566407.

[41] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, D. Garant, Botnet detection based on traffic behavior analysis and flow intervals, Computers & Security 39, Part A (0) (2013) 2–16. doi:http://dx.doi.org/10.1016/j.cose.2013.04.007.
URL http://www.sciencedirect.com/science/article/pii/S0167404813000837

[42] D. il Jang, K. yu Cho, M. Kim, H. chul Jung, B.-N. Noh, Evasion technique and detection of malicious botnet, in: International Conference for Internet Technology and Secured Transactions (ICITST), 2010, pp. 1–5.

[43] M. Masud, T. Al-Khateeb, L. Khan, B. Thuraisingham, K. Hamlen, Flow-based identification of botnet traffic by mining multiple log files, in: First International Conference on Distributed Framework and Applications, 2008, pp. 200–206. doi:10.1109/ICDFMA.2008.4784437.

Journal Pre-proof

[44] I. H. Witten, E. Frank, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, 2005.

[45] D. Dittrich, S. Dietrich, P2p as botnet command and control: A deeper insight, in: 3rd International Conference on Malicious and Unwanted Software, 2008, pp. 41–48. doi:10.1109/MALWARE.2008.4690856.

[46] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, D. Dagon, Peer-to-peer botnets: overview and case study, in: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, USENIX Association, 2007, pp. 1–1.

[47] K.-S. Han, K.-H. Lim, E.-G. Im, The traffic analysis of p2p-based storm botnet using honeynet, Journal of the Korea Institute of Information Security and Cryptology 19 (4) (2009) 51–61.

[48] S.-K. Noh, J.-H. Oh, J.-S. Lee, B.-N. Noh, H.-C. Jeong, Detecting p2p botnets using a multi-phased flow model, in: Third International Conference on Digital Society, 2009, pp. 247–253. doi:10.1109/ICDS.2009.37.

[49] S. Zihao, W. Hui, Network data packet capture and protocol analysis on jpcap-based, in: International Conference on Information Management, Innovation Management and Industrial Engineering, Vol. 3, 2009, pp. 329–332. doi:10.1109/ICIII.2009.388.

[50] A. Hegna, Visualizing spatial and temporal dynamics of a class of irc-based botnets.
URL http://ntnu.diva-portal.org/smash/record.jsf?pid=diva2:353050

[51] M. Alauthaman, N. Aslam, L. Zhang, R. Alasem, M. A. Hossain, A p2p botnet detection scheme based on decision tree and adaptive multilayer neural networks, Neural Computing and Applications 29 (11) (2018) 991–1004. doi:10.1007/s00521-016-2564-5.
URL https://doi.org/10.1007/s00521-016-2564-5

Journal Pre-proof

[52] M. Roughan, S. Sen, O. Spatscheck, N. Duffield, Class of service mapping for qos: A statistical signature-based approach to ip traffic classification, in: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, IMC 04, ACM, New York, USA, 2004, pp. 135–148. doi:10.1145/1028788.1028805.

[53] H. T. Nguyen, S. Petrović, K. Franke, A comparison of feature-selection methods for intrusion detection, in: Computer Network Security, Springer, 2010, pp. 242–255. doi:10.1007/978-3-642-14706-7_19.

[54] M. Alkasassbeh, An empirical evaluation for the intrusion detection features based on machine learning and feature selection methods, Journal of Theoretical and Applied Information Technology 95.

[55] C. Livadas, R. Walsh, D. Lapsley, W. Strayer, Usilng machine learning technliques to identify botnet traffic, in: Proceedings 31st IEEE Conference on Local Computer Networks, 2006, pp. 967–974. doi:10.1109/LCN.2006.322210.

[56] P. Van der Putten, M. Van Someren, A bias-variance analysis of a real world learning problem: The coil challenge 2000, Machine Learning 57 (1-2) (2004) 177–195. doi:10.1023/B:MACH.0000035476.95130.99.
URL http://dx.doi.org/10.1023/B%3AMACH.0000035476.95130.99

[57] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, Classification and regression trees, CRC press, 1984.

[58] T.-F. Yen, Detecting stealthy malware using behavioral features in network traffic, Thesis (2011).

[59] D. Rossi, E. Sottile, P. Veglia, Black-box analysis of internet p2p applications, Peer-to-Peer Networking and Applications 4 (2) (2011) 146–164. doi:10.1007/s12083-010-0072-4.
URL http://dx.doi.org/10.1007/s12083-010-0072-4

Journal Pre-proof

[60] M. Scanlon, T. Kechadi, Peer-to-Peer Botnet Investigation: A Review, Vol. 179 of Lecture Notes in Electrical Engineering, Springer Netherlands, 2012, book section 33, pp. 231–238, (Jong Hyuk). doi:10.1007/978-94-007-5064-7_33.
URL http://dx.doi.org/10.1007/978-94-007-5064-7_33

[61] Dshield.org, Most attacked port reports (2013).
URL http://www.dshield.org/portreport.html

[62] T. Limmer, F. Dressler, Flow-based tcp connection analysis, in: IEEE 28th International Conference on Performance Computing and Communications (IPCCC), 2009, pp. 376–383. doi:10.1109/PCCC.2009.5403846.

[63] T. M. Cover, J. A. Thomas, Elements of information theory, John Wiley & Sons, 2012.

[64] M. Jaber, R. Cascella, C. Barakat, Can we trust the inter-packet time for traffic classification?, in: IEEE International Conference on Communications (ICC), 2011, pp. 1–5. doi:10.1109/icc.2011.5963024.

[65] R. Gusella, Characterizing the variability of arrival processes with indexes of dispersion, IEEE Journal on Selected Areas in Communications 9 (2) (1991) 203–211. doi:10.1109/49.68448.

[66] A. Nigrin, Neural networks for pattern recognition, MIT press, 1993.

[67] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, W.-Y. Lin, Intrusion detection by machine learning: A review, Expert Systems with Applications 36 (10) (2009) 11994–12000. doi:doi:http://dx.doi.org/10.1016/j.eswa.2009.05.029.
URL http://www.sciencedirect.com/science/article/pii/S0957417409004801

[68] M. A. Razi, K. Athappilly, A comparative predictive analysis of neural networks (nns), nonlinear regression and classification and regression tree

Journal Pre-proof

(cart) models, Expert Systems with Applications 29 (1) (2005) 65–74. doi:doi:http://dx.doi.org/10.1016/j.eswa.2005.01.006.
URL http://www.sciencedirect.com/science/article/pii/S0957417405000072

[69] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the rprop algorithm, in: IEEE International Conference on Neural Networks, 1993, 1993, pp. 586–591 vol.1. doi:10.1109/ICNN.1993.298623.

[70] Z. Boger, H. Guterman, Knowledge extraction from artificial neural network models, in: IEEE International Conference on Systems, Man, and Cybernetics, Vol. 4, 1997, pp. 3030–3035 vol.4. doi:10.1109/ICSMC.1997.633051.

[71] K. Gai, M. Qiu, Reinforcement learning-based content-centric services in mobile sensing, IEEE Network 32 (2018) 34–39.

[72] Barto, Andrew, Reinforcement learning: An introduction, Cambridge: MIT Press, 1998.

[73] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, P. Hakimian, Detecting p2p botnets through network behavior analysis and machine learning, in: Ninth Annual International Conference on Privacy, Security and Trust (PST), 2011, pp. 174–180. doi:10.1109/PST.2011.5971980.

[74] A. Shiravi, H. Shiravi, M. Tavallaee, A. A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Computers & Security 31 (3) (2012) 357–374. doi:http://dx.doi.org/10.1016/j.cose.2011.12.012.
URL http://www.sciencedirect.com/science/article/pii/S0167404811001672

Journal Pre-proof

[75] F. K. Aaron Turner, Tcpreplay version 4.0.0 (2013).
URL http://tcpreplay.appneta.com

[76] J. Espinosa, J. Vandewalle, Constructing fuzzy models with linguistic integrity from numerical data-afreli algorithm, Fuzzy Systems, IEEE Transactions on 8 (5). doi:10.1109/91.873582.

[77] B. W. Matthews, Comparison of the predicted and observed secondary structure of t4 phage lysozyme, Biochimica et Biophysica Acta (BBA) - Protein Structure 405 (2) (1975) 442–451. doi:http://dx.doi.org/10.1016/0005-2795(75)90109-9.
URL http://www.sciencedirect.com/science/article/pii/0005279575901099

[78] J. A. Swets, Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers, Psychology Press, 2014.

[79] T. Fawcett, An introduction to roc analysis, Pattern Recognition Letters 27 (8) (2006) 861–874. doi:10.1016/j.patrec.2005.10.010.
URL http://www.sciencedirect.com/science/article/pii/S016786550500303X

[80] S. W. Timothy, L. David, W. Robert, L. Carl, Botnet Detection Based on Network Behavior, Vol. 36 of Advances in Information Security, Springer US, 2008, book section 1, pp. 1–24. doi:10.1007/978-0-387-68768-1_1.
URL http://dx.doi.org/10.1007/978-0-387-68768-1_1

[81] G. Gu, P. A. Porras, V. Yegneswaran, M. W. Fong, W. Lee, Bothunter: Detecting malware infection through ids-driven dialog correlation, in: USENIX Security, Vol. 7, 2007, pp. 1–16.

[82] G. Gu, R. Perdisci, J. Zhang, W. Lee, Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection, in: USENIX Security Symposium, 2008, pp. 139–154.

Journal Pre-proof

[83] G. D. L. T. Parra, P. Rad, K. R. Choo, Implementation of deep packet inspection in smart grids and industrial internet of things: Challenges and opportunities, J. Network and Computer Applications 135 (2019) 32–46.

[84] K. Wang, C.-Y. Huang, S.-J. Lin, Y.-D. Lin, A fuzzy pattern-based filtering algorithm for botnet detection, Computer Networks 55 (15) (2011) 3275 – 3286. doi:http://dx.doi.org/10.1016/j.comnet.2011.05.026.

[85] L. Al Shalabi, Z. Shaaban, Normalization as a preprocessing engine for data mining and the approach of preference matrix, in: International Conference on Dependability of Computer Systems, 2006, pp. 207–214. doi:10.1109/DEPCOS-RELCOMEX.2006.38.

[86] A. Dries, U. Rückert, Adaptive concept drift detection, Statistical Analysis and Data Mining 2 (5-6) (2009) 311–327. doi:10.1002/sam.10054.
URL http://dx.doi.org/10.1002/sam.10054

Journal Pre-proof

Journal Pre-proof

Mohammed Alauthman received his PhD degree from Northumbria University Newcastle, UK in 2016. He received a B.Sc. degree in Computer Science from Hashemite University, Jordan, in 2002, and received M.Sc. degrees in Computer Science from Amman Arab University, Jordan, in 2004. Currently, he is Assistant Professor and senior lecturer at department of computer science, Zarqa University, Jordan. His research interests include cyber-security, Cyber Forensics, advanced machine learning and data science applications.

Nauman Aslam is a Reader in the Department of Computer Science and Digital Technologies. He joined Northumbria University in August 2011. Dr. Nauman received his PhD in Engineering Mathematics from Dalhousie University, Halifax, Nova Scotia, Canada in 2008. Prior to joining Northumbria University he worked as an Assistant Professor at Dalhousie University, Canada from 2008 - 2011. Currently, he also holds an adjunct assistant professor position at Dalhousie University

Mouhammd Alkasassbeh graduated from the school of computing, Portsmouth University, UK in 2008. He is currently a full professor in the Computer Science Dept. Princess Sumaya University for Technology. His research interests include Network Traffic Analysis, Network Fault Detection, Classification Network Fault and abnormality and Machine learning in the area of computer networking and network security.

SULEMAN KHAN received the Ph.D. degree (Hons.) from the Faculty of Computer Science and Information Technology, University of Malaya, Malaysia, in 2017. He is a Faculty Member with the School of Information Technology, Monash University Malaysia Campus. He has published more than 45 high-impact research articles in reputed international journals, including the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, ACM Computing Surveys, the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. He has published in the 2018 Local Computer Networks Conference. His research areas include, but are not limited to, network forensics, software-defined networks, the Internet of Things, cloud computing, and vehicular communications

Ahmad Al-Qerem graduated in applied mathematics and MSc in Computer Science at the Jordan University of Science and Technology in 1997 and 2002, respectively. After that, he was appointed as full-time lecturer at the Zarqa University. Currently he is a visiting professor at Princess Sumaya University for Technology (PSUT). He obtained a PhD from Loughborough University, UK. His research interests are in performance and analytical modeling, mobile computing environments, protocol engineering, communication networks, transition to IPv6, and transaction processing. He has published several papers in various areas of computer science. Currently, he has a full academic post as associate professor and the head of the Department of Internet Technology at Zarqa University - Jordan.

Kim-Kwang Raymond Choo (SM15) received the Ph.D. in Information Security from Queensland University of Technology. He currently holds the Cloud Technology Endowed Professorship at the University of Texas at San Antonio. In 2016, he was named the Cybersecurity Educator of the Year - APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn), and in 2015 he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He is the recipient of the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty, ESORICS 2015 Best Paper Award, 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, Fulbright Scholarship in 2009, 2008 Australia Day

Journal Pre-proof

Achievement Medallion, and British Computer Society's Wilkes Award in 2008. He is also a Fellow of the Australian Computer Society, an IEEE Senior Member, and Co-Chair of IEEE Multimedia Communications Technical Committee (MMTC)'s Digital Rights Management for Multimedia Interest Group.

Journal Pre-proof