A Novel Machine Learning-Based Approach for the Detection of SSH Botnet Infection

José Tomás Martínez Garre[a], Manuel Gil Pérez[*,a], Antonio Ruiz-Martínez[a]

[a] *Department of Information and Communications Engineering, University of Murcia, 30100 Murcia, Spain*

Abstract

Botnets are causing severe damages to users, companies, and governments through information theft, abuse of online services, DDoS attacks, etc. Although significant research is being made to detect them and mitigate their effect, they are exponentially increasing due to new zero-day attacks, a variation of their behavior, and obfuscation techniques. High Interaction Honeypots (HIH) are the only honeypots able to capture attacks and log all the information generated by attackers when setting up a botnet. The data generated is being processed using Machine Learning (ML) techniques for detection since they can detect hidden patterns. However, so far, research has been focused on intermediate phases of the botnet's life cycle during operation, underestimating the initial phase of infection. To the best of our knowledge, this is the first solution in the infection phase of SSH-based botnets. Therefore, we have designed an approach based on an SSH-based HIH to generate a dataset consisting of executed commands and network information. Herein, we have applied ML techniques for the development of a real-time detection model. This approach reached a very high level of prediction and zero false negatives. Indeed, our system detected all known and unknown SSH sessions intended to infect our honeypots. Thus, our research has demonstrated that new SSH infections can be detected through ML techniques.

Key words: botnet, machine learning, zero-day malware, honeypot, high interaction

[*]Corresponding author. Phone: +34 868 887645, Fax: +34 868 884151.

Email addresses: josetomas.martinez@um.es (José Tomás Martínez Garre), mgilperez@um.es (Manuel Gil Pérez), arm@um.es (Antonio Ruiz-Martínez)

Work submitted to Future Generation Computer Systems

1. Introduction

Botnets continue to pose a high risk to governments, industries, companies, and individual users. Indeed, its number has been increasing annually: Spamhaus Malware Labs identified 17,602 botnet Command and Control (C&C) servers in 2019, which means an increment of 71.5% in the botnet C&Cs number compared to the previous year [1]. Among recent botnet attacks, it is estimated that the 3ve botnet has been the cause of $19 billion in advertising theft, involving the orchestration of about 1.7 million PCs through this malware [2]. Additionally, botnets have taken advantage of the significant increase of Internet of Things (IoT) popularity, together with its weak security level, to turn IoT devices into a great and powerful platform for cyberattacks, especially for Distributed Denial-of-Service (DDoS) attacks [3]. Hence, in 2016, the French web host and cloud service provider OVH underwent a 1.1 Tbps DDoS attack by a botnet created by the malware Mirai [4], composed mainly of surveillance cameras and video recorders [5]. Later, in 2018, cybersecurity researchers discovered Chabulo (ChaCha-Lua-bot), which is a new botnet that launches DDoS attacks through compromised SSH servers and IoT devices [6]. Recently, in 2019, Trend Micro's honeypot security systems detected a URL spreading a botnet with a Monero miner bundled with a Perl-based backdoor component to perform DDoS attacks [7]. All these examples of botnets have in common that they have used as attack vector the Secure Shell (SSH) remote access service, which is a common attack vector for IoT devices [8]. Therefore, we decided to focus our research on botnets that use SSH as an attack vector.

Each botnet is composed of many malware-infected computers (bots), which are under human control (botmaster) through C&C servers [9, 10, 11]. The variety of malware used is increasing and with different behavior. Indeed, as reported by the AV-TEST Institute[1], 350,000 new malware samples are registered every day. The exponential increase in zero-day malware or malware variants, which modify their behavior as they act, and use of advanced obfuscation techniques help the malware to evade the traditional signature-based and heuristics-based detection techniques [12].

[1] https://www.av-test.org/

A practical method for detecting new malware is to use honeypots and analyze the log events they generate to reveal intelligence information and malware behavior on current threats. Honeypot systems are designed to capture attacks by simulating real services and applications, which have been configured dynamically according to the environment expected by malware. However, only High Interaction Honeypots (HIH) can capture all phases of a botnet, which are *infection*, *communication*, and *attack* [13]. Low and medium interaction honeypots offer a vast limitation for running executable files, so it would be challenging to capture logs from files for bots infection.

Nowadays, as aforementioned above, the number of attacks is dramatically growing, and analyzing the massive amount of log events generated by honeypots is a challenge for security analysts. Therefore, new analysis techniques are being developed, where Machine Learning (ML) is playing a crucial role.

Additionally, it is believed that bots, in any of their phases, have hidden behavior patterns in the activities they perform [14]. This hypothesis, together with the appearance and constant need to detect known or unknown malware and to analyze a large amount of data, leads to the current trend of developing bot detection methods based on ML techniques. These techniques detect unknown malware by getting knowledge from ones previously detected, being able to generate fewer false positives and false negatives compared to other malware detection methods [15]. Solutions to learn how to tackle zero-day attacks, as the one proposed in [16], are encouraging the application of transfer learning techniques to train the model in a dataset and test it later by using another unrelated dataset. These techniques can also be applied to adapt honeypot behaviors according to the actions of the attacker and, therefore, the honeypot is more difficult to detect [17].

Due to ML can be used to detect hidden patterns, several ML solutions have been defined for botnet detection. Most of these solutions are focused on identifying bots in the communication and attack phases [18, 19, 20, 21]. There are very few solutions proposed for the detection of the infection process of a botnet, in particular, Telnet-based botnets [22, 23]. Due to security flaws, the Telnet protocol has been replaced by the SSH one in scenarios where security was required. However, to the best of our knowledge, there is no solution focused on the detection of the infection process in SSH-based botnets. The detection process in the communication and attack phases can suppose specific problems and risks since the bot could have already produced some damage. However, a detection during the infection phase would have

the advantage that the bot can be identified and disabled before participating in any malicious activity.

To address this issue, we decided to focus our research on the development of a novel approach that used ML to detect the infection phase through the SSH service performed by a known or unknown botnet. To this end, features extracted from the behavior captured by the honeypots are used, such as the executed commands and the network traffic statistics generated in each SSH session.

For this purpose, our solution is based on ML using Supervised Learning algorithms to automate the detection and prediction of incoming SSH security threats aiming to infect new devices. Thus, the approach presented here requires evaluated and tagged data to build the model and classify the SSH sessions' behavior, depending on whether they intend to infect the victim or not, to speed-up the learning and detection processes. Then, the ML-based model is trained for the detection of the SSH botnets' infection phase. To the best of our knowledge, there are no public datasets formed by executed commands and network traffic statistics generated during an SSH session. Therefore, we decided to use a high interaction SSH honeypot (HIH) to create a dataset for the effective and dynamic training of our ML model.

To design and develop this new novel approach, we established the following goals:

- To design and implement an ML-based approach for the detection of an SSH botnet infection in real-time, reducing the risk/possibility of a future attack.
- To create a novel dataset composed of executed commands and network features generated during an SSH session.
- To evaluate the ML algorithms used for the most known error metrics.
- To develop a model capable of identifying SSH zero-day malware, pretending to infect a new device.
- To assess the importance of the features extracted from the behavior of bots during the process of infection.

The remainder of the paper is structured as follows. Section 2 describes background information on botnets, honeypots, and given ML models used in

the paper, while Section 3 discusses some related work on botnets and how ML has been useful for their detection. Section 4 describes our proposed approach and how it works to detect incoming SSH security threats. Next, Section 5 shows the proposed dataset and detection model. Section 6 presents the results and findings of our research. Finally, conclusions and future work are drawn in Section 7.

2. Background

Along this section, we introduce what a botnet is, the different phases a botnet has, and we detail the infection phase of SSH-based botnets. Next, the importance of honeypots in botnet research is shown. Finally, we briefly describe what Machine Learning is and related terms discussed in this paper.

2.1. Botnets

As previously introduced, a botnet is a network composed of bots that have been infected with malware and that are controlled by individual hackers, hackers groups, or Government/Nation-state Actors [24]. Their malicious intention can be to perform a fraudulent and malicious online activity such as information theft, abuse of online services, DDoS attacks, dissemination of spam, crypto, click-fraud, and service disruption [9, 10, 11, 24].

The life cycle of a botnet is composed of the phases of infection, communication, and attack [13]. Other authors, as in [9], divide the life cycle into more phases: initial injection, secondary injection, connection or rally, malicious activities, and maintenance and upgrading. As both life cycles can be mapped, for the sake of simplicity, we have opted for the most general model defined by [13], where phases are differentiated only by the communication type and their primary objective. Next, we present a brief and general description of these phases, considering that these phases could vary depending on the type of device: desktop computer, laptop, IoT device, etc.

In the *infection phase*, the computer is infected with malware actively or passively (e.g., SSH service or email, respectively), through different attack vectors, which are used for downloading the malware binaries that will turn the victim into a potential bot. Secondly, in the *communication phase*, communications take place between the infected computers and the C&C servers. This communication is performed for two reasons: to become a new member of the botnet and to update its behavior. Lastly, in the *attack phase*, the

bots are the ones that carry out the malicious activity through instructions received from the botmaster.

We have provided a general view of a botnet life cycle. Next, we give more details on the infection process, considering that the attacked device has an SSH service since this is where our work is focused. These devices can only be infected actively.

In the infection process, SSH sessions tend to follow a set of steps that depends on the botnet family they belong to [4, 25]. An example, Marinho and Holanda [25] make use of a high interaction SSH honeypot infected by the Rakos botnet. In their investigation, the Rakos botnet follows different steps: (1) it performs brute-force attack to the SSH service; (2) it collects device information; (3) it validates that it has permissions; (4) it loads the malicious binary; (5) it executes malware; (6) it removes signs of attack; and (7) it searches for new SSH victims. This pattern is usual in SSH botnets [26], but because variants and new forms are continually appearing, new techniques able to recognize zero-day attacks are needed. Furthermore, as pointed out by Sadasivam et al. [27], the brute-force attack to SSH service is made using a botnet. In 2019, Outlaw Hacking Group's Botnet gained access to the SSH servers through a brute-force attack and downloaded a shell script. The shell script downloads, extracts, and executes the payload. The extracted TAR file contained folders with scripts, a miner, and backdoor components [7].

2.2. Honeypots

A handy tool to obtain intelligence information about a bot infection behavior is honeypots [26, 27, 28]. Honeypots are network resources used to attract non-legitimate users who try to compromise a system with vulnerable machines and services, although they are virtualized assets mounted by the honeypot to go unnoticed for such users. Any interaction between the attacker and the honeypot is captured for further analysis. Honeypots are widely used in research [17, 29], where a considerable amount of information is retrieved to design new detection strategies and identify zero-day attacks.

Three types of honeypots can be distinguished according to the level of interaction that the honeypot allows the attacker [28]: low, medium, and high. Low interaction honeypots offer limited interaction between the system and the attacker. The purpose is to detect and save unauthorized connections. These honeypots are easy to maintain, and the added risk to the network is shallow. But their ability to capture data is limited, and they are easy to

detect by the attacker. Medium interaction honeypots provide more significant interaction with the attacker and allow the simulation of a service or an operating system where everything is monitored. In this type of honeypot, the attacker could carry out some tasks, such as the execution of specific commands and download files. Other functions, like binary execution, are forbidden. Higher information-gathering capabilities are achieved, but this has a higher risk. Finally, high interaction honeypots are computers with a real operating system and vulnerable services, whose aim is to attract the attacker and analyze his/her behavior inside the host. These honeypots involve a high risk because the attacker could take full control of the host. Furthermore, these honeypots are challenging to maintain and would require additional security, such as firewall rules that restrict outgoing connections.

Low and medium interaction honeypots offer a significant limitation when capturing all the interaction produced by botnets since the infection of a new bot is usually done by executing a binary or script, and both types of honeypot do not allow it.

2.3. Machine Learning

Nowadays, cybersecurity experts use ML-based tools to support, but not to replace, existing traditional methods to improve malicious activity detection, human analysis, incident response automation, and identification of zero-day exploits, among others [30]. Machine Learning [31] is an artificial intelligence field that aims to build and to study systems with the ability to learn from data. These systems can recognize complex patterns and make qualified decisions based on experience. ML algorithms are broadly categorized into two, namely: supervised and unsupervised machine learning algorithms. We provide below a brief overview of them since these algorithms are used in some solutions that we will present in Section 3 on the related work.

Supervised Learning (SL) builds models, from tagged data that have been evaluated previously, that map inputs to desired outputs. The honeypot will learn from the patterns contained in the labeled data to classify (i.e., predict) labels for new, unseen data. In the context of botnet detection, SL algorithms are used in network traffic classification [11] and malware detection [32]. For example, network classifiers can sort network traffic as malicious or non-malicious as well as identify traffic belonging to different botnets. Popular SL algorithms used in botnet detection include Decision Tree, Random Forest,

SVM, and Naive Bayes. We used in this research these algorithms. A brief description of them is shown below [33]:

- *Decision Tree (DT)* is a model that makes predictions by posing a series of simple tests on the given point. This process can be represented as a tree, where its leaves are the values to be predicted. DT has less requirement of data cleaning compared to other algorithms. Instead, DT may have an overfitting issue, which can be resolved using Random Forest, and the computational complexity may increase for more class labels.

- *Random Forest (RF)* is a specific ensemble method where the individual models are decision trees trained in a randomized way to reduce correlation among them. RF takes a low training time, predicts output with high accuracy, and provide a reliable feature importance estimate. Instead, RF is inherently less playable than an individual DT, and training a large number of deep trees can have high computational costs.

- *Support Vector Machine (SVM)* is a model aiming to find the best decision boundary, or hyperplane, that separates n-dimensional space into different classes. Its objective is to find a plane that has the maximum margin, which means that future data points can be classified with more confidence. SVM works fine with a clear margin of separation between classes and is useful in high dimensional spaces. Instead, SVM requires high time training when we have a large dataset.

- *Naive Bayes (NB)* is a probabilistic model based on the Bayes' theorem. This model is fast and easy to implement, but its most significant disadvantage is the requirement of predictors to be independent. In most of the real-life cases, the predictors are dependent, and this hinders the performance of the classifier.

Unsupervised Learning (UL) is modeling the underlying or hidden structure or distribution in the data to learn more about the data. This method does not need labeled data and does not need to be trained beforehand. UL algorithms allow discovering groups of similar examples within the input data, where it is called clustering, to determine the distribution of data within the input space, known as density estimation. In botnet detection,

UL algorithms are commonly used for the clustering of bot-related behaviors. The algorithms most popularly used in botnet detection are K-means, X-means, and hierarchical clustering.

The most commonly used error metrics are:

- *Accuracy*, referring to what percent of correct predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

- *Precision*, indicating what percent of positive predictions were correct.

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

- *Recall* or *sensitivity*, defining what percent of positive cases did a classifier catch.

$$Recall = \frac{TP}{TP + FN}$$

- *F1 score*, showing the trade-off between the precision and recall as fas the positive class is concerned.

$$F1score = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

And error graphics are:

- *ROC Curve*, summarizing the trade-off between the true positive rate and the false positive one for our predictive model using different probability thresholds.

- *Precision-Recall Curve*, summarizing the trade-off between the true positive rate and the positive predictive value for our predictive model using different probability thresholds.

Before training the ML model, it is necessary to select features from a dataset that better describe the attacker's behavior such as API calls, network traffic, executed commands, system changes, etc. A selection of features is required to extract an optimal subset of features that best represent the data. Datasets with many irrelevant features can lead the model into an

underfitting state, where the model does not fit the training data and therefore misses the underlying trend of the data. Some of the most frequently features used in botnet detection are [34]: connection time, length of the first transmitted packet, the total number of sent and received bytes, and the ratio between the number of incoming and outgoing packets.

3. Related Work

Currently, a large number of studies focus on the detection of botnets using ML techniques. These techniques have been mainly used for the identification of malicious traffic flows, or from preventing the misidentification of non-malicious traffic flows [11]. Herein, a summary of the most significant works to date is shown.

In 2006, Livadas et al. [35] introduced one of the first research demonstrating the possibility of using ML for botnet detection. This study focused on detecting the communication phase of IRC botnets using supervised learning to classify IRC bot with statistical features from the TCP traffic flow. Their assessment showed the importance of the features' selection in botnet detection.

In 2012, Shin et al. [36] developed an SVM-based botnet detection method for host and network-based analysis. In this research, the system was able to detect all the existing bots in the tests, which were in communication and attack phases.

In 2014, Stevanovic et al. [37] presented a flow-based botnet detection in the communication phase using RF classifier. This approach used a set of statistical flow features to train its detection model. This detection model reached a sound assessment for a limited number of packets and a limited duration of time of monitoring per flow.

In 2016, Kirubavathi et al. [38] performed a study on ML-based botnets detection via mining of network traffic flow behavior, classifying the traffic into bot and normal traffic. This approach focused on identifying botnets, regardless of their structures during the communication phase.

In 2018, Wu et al. [39] presented a bot detection approach using clustering techniques during communication and attack phases.

In 2019, Khan et al. [40] developed a Supervised Learning-based technique to detect P2P botnets in communication and attack phases. Also, in 2019, Pektaş et al. [19] presented a deep neural network-based approach to detect botnet by modeling network traffic traces between communication endpoints

by using TCP, UDP, and ICMP flow features. This approach focused on identifying botnets' communication and attack phases.

Finally, in 2020, Wang et al. [20] showed a hybrid botnet detection approach based on the analysis of flow-based and graph-based traffic behaviors during communication and attack phases. Lastly, Biradar et al. [21] presented a Supervised Machine Learning approach for botnets detection using DNS query data. The authors claimed that they could identify the presence of bots, in the communication and attack phases, by checking DNS requests.

The studies discussed above focus on the communication and attack phases. We have found very few studies about the infection phase. In 2018, Bajtoš et al. [22] made a study about bots' behavior during the infection phase via Telnet. Note that no ML techniques were used in this approach. By using a set of Telnet-based honeypots, they distinguished 9 botnet families according to observed properties during the infection of the honeypot. The authors examined the IP address in which the compromised host connects to download the malicious binary, what way the attacker uses to download the malicious binary and where it is downloaded. The results showed that each botnet family uses a particular range of IP addresses to compromise the honeypot and another specific range of IP addresses to download the malicious binaries. The most used download methods are wget, tftp, curl, and ftpget commands. As an alternative to downloading, the attacker uses echo command to write shellcode to a file for later run. Also, the results show a high correlation between commands used to download a binary, change its permissions, run it, and delete it.

This last research work was continued in [23], where the authors designed a model to profile threat agents in Telnet-based botnet groups. By using clustering through PAM and K-modes algorithms, they were able to identify different specific features in several types of botnets. These features were the number of sessions and the number of credential guesses. Depending on them, the authors claimed that they could predict the next behavior of attackers, determined by the cluster to which the botnet belongs.

Table 1 presents a summary of the discussed works. As shown, the detection of botnets has been performed with supervised learning, unsupervised learning, and deep learning algorithms, where all approaches present good results. The features used for these algorithms were mainly extracted from the network traffic generated by the bots and the behavior observed at the endpoint. One aspect to highlight is regarding the botnet's detection phase. These studies are mostly focused on detecting bots within phases 2 and 3,

i.e., the communication and attack phases, respectively. Very few approaches have been made with ML techniques that work on the infection phase of a bot's life cycle and, on the other hand, no approach focused on that first phase makes its proposals through exploited vulnerabilities to the SSH service. The main advantage of being able to detect a bot in its first phase is that it can be disabled before it participates in any possible future attack.

Ref.	Learning	Detection Phase	Features	Assessment
[35]	DT, NB, and BN	2	TCP flow	FPR: 10-20%, FNR: 30-40%
[36]	SVM	2	Host and Network-based	TPR: 100%, FPR: $< 1\%$
[37]	RF	2	Flow statistics	F1 score: 94.83%
[38]	DT, NB, and SVM	2, 3	Flow statistics	Accuracy: 99.14%, F1 score: 96.9%
[39]	K-Means, X-Means, and EM	2, 3	Network-based	Accuracy: 95.15%
[40]	NB, DT, and ANN	2, 3	Network-based	Accuracy: 94.4%
[19]	Deep Neural Network	2, 3	TCP, UDP, and ICMP flow	Accuracy: 99.3%, F1 score: 99.1%
[20]	K-means, LSM, and LOF	2, 3	Flow and graph-based	Accuracy: 99.94%, FPR: 0.06%
[21]	SVM	2, 3	DNS query data	Accuracy: 91.80%, F1 score: 91.8%
[23]	PAM and K-modes	1	Host-based	*Unknown*

Table 1: State of the art in detecting bots using Machine Learning techniques.

4. Our approach

In our proposed solution, we focus on detecting the infection process of an SSH botnet and, particularly, the variants and new methods yet unknown, not identified by signature-based detection systems. As presented in Section 3, ML techniques seem to be ideal for our detection system due to: the large size of data that it can analyze in real-time; the ability to learn from experience; the existence of hidden patterns in the activities performed by bots; and the need to deal with unknown malware. Thus, to do this, we have used a honeypot architecture with an ML-based detection system.

As shown in Figure 1, the different entities that participate in this scenario are the attacker, the honeypots with the SSH sensors, and the ML-based detection system. The flow of the process for the proposed solution starts when an attacker attempts to inject the malware through port 22, default SSH port, by logging into a device through a brute-force or dictionary attack. In this context, malicious SSH behavior performed by the attackers is captured by our SSH sensors. After the attack, the SSH sensor sends the captured intelligence information and its self-internal state to the ML-based detection system, where the records received by all SSH sensors are stored

in a database. Next, our system generates a new ML-based infection detection model according to the records stored in our database. Finally, the new model is sent to the SSH sensors so that they can update and improve their previous detection model. This process is repeated every day automatically.

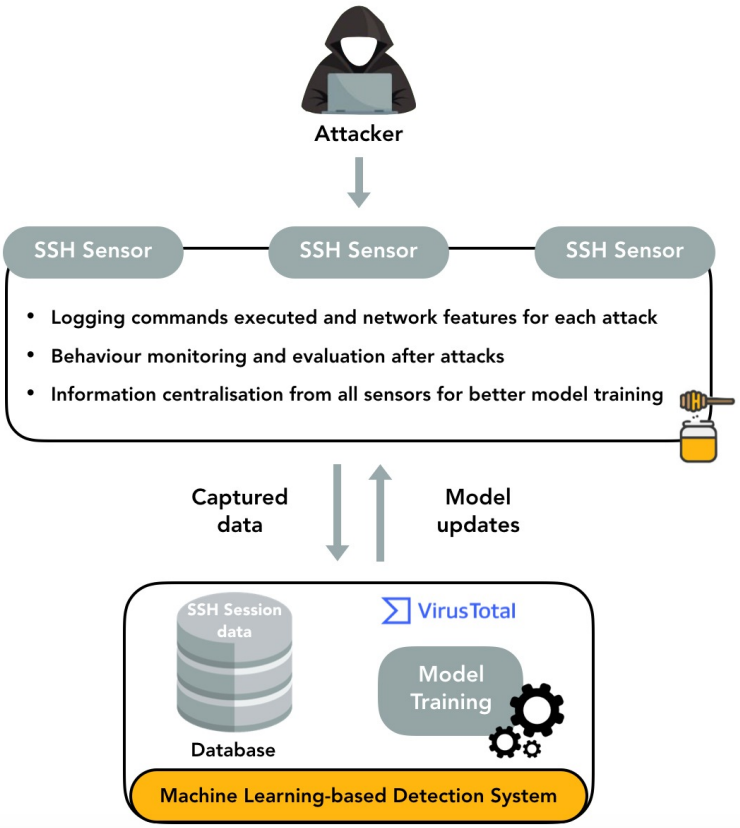


Figure 1: The ML-based detection framework proposed for the botnet's infection phase.

Next, we present how we have developed the elements of our architecture, that is, the honeypots and the ML detection system as well as its design details. Specifically, our sensors are based on HonSSH [41], which is a high interaction SSH honeypot that logs all SSH sessions with the attacker. The honeypots act as a proxy between the attacker and an SSH server, creating two separate SSH connections between them. Each sensor can keep several SSH sessions open, with independent systems through Docker containers. The sensor gathers information regarding the commands executed and net-

work traffic. Besides, the associated container must be kept active for some time after the SSH session is closed, aiming at identifying if the host (honeypot) has been infected during the session execution. It will facilitate the labeling of the SSH sessions in the dataset. Once the attacker logs out of the server, our honeypot automatically installs a dynamic rule in the firewall to block all outgoing traffic from the container associated with this session. This fact will prevent our infected computers from being part of an attack. These sensors will send all the information gathered to a centralized database.

Our honeypots are configured to allow unlimited access. Our goal is to capture all commands executed by the attacker and network traffic statistics during an SSH session. Executed commands by the attacker can provide us a lot of information about the purpose of that SSH session. However, training our system only with the executed commands could give unreliable results, since several commands are performing the same operation, e.g., wget and curl to download an executable file. Network statistics have also been used to solve this limitation. This way, commands with different syntax executed in SSH sessions, but with the same goal, will be able to reveal similar SSH session exposing hidden patterns during the infection phase. Table 2 describes the network features, based on previous ML-based botnet detection researches, used in our system.

Feature	Description
session_duration	SSH session duration in seconds.
ips_src_count	Different source IPs for incoming connections.
ips_dst_count	Different destination IPs for outgoing connections.
urls	URLs in use.
sent_bytes	Bytes sent by the attacker.
received_bytes	Bytes received by the Honeypot.
tcp_bytes	TCP bytes transmitted in the SSH session.
udp_bytes	UDP bytes transmitted in the SSH session.
size_payloads	Payload bytes transmitted in the SSH session.
size_first_packet	First packet's length transmitted in the SSH session.
max_size_sent_packet	Sent packet's maximum size.
min_size_sent_packet	Sent packet's minimum size.
max_size_received_packet	Received packet's maximum size.
min_size_received_packet	Received packet's minimum size.

Table 2: Network features to model attacker behavior.

The ML-based detection (MLD) system is responsible for obtaining all the SSH session information from the database to classify it and generates a model that represents it.

All the captured information was used to build the dataset that trained our system. MLD system selects the most optimal features; therefore, those that best model the behavior of the attackers. Also, our detection model continues to train continually as the dataset increases, thus achieving more accurate results. The entire model training process requires high computational capabilities.

Due to the low processing capacity and memory of the sensors, MLD system is in charge of performing the training of the model in a centralized way, and the sensors only worry about reporting the SSH session information and dynamically updating their detection model.

Our system works with supervised learning algorithms and, as such, needs to tag data. Supervised learning has been selected because our system must initially learn from the attacks received previously, achieving a better detection model. This trained model will be able to detect known and unknown incoming SSH security threats. Session tagging is performed automatically, with two possible options: SSH sessions that do not infect the machine (value 0 or negative class) or SSH sessions that have the purpose of recruiting a new bot (value 1 or positive class).

Correct labeling of the dataset is essential for an accurate evaluation of results, which is achieved thanks to our honeypots. They are capable of detecting if there is a non-legitimate activity within the honeypot once the SSH session has been closed. Any action after closing the session is considered non-legitimate activity. This situation would indicate that our honeypot has been infected. Therefore, our system labels the sessions as positive in case it detects any activity in the container after the session is closed. This utility is only useful to add new SSH sessions to our dataset and thus improve our detection model. It could never be achieved with real equipment, as it will have legitimate activity.

Besides, the labeling mechanism is supported by VirusTotal[2]. This system relies on the VirusTotal platform to upload and scan files or URLs to determine if it is malware or not. In case any file was downloaded during that session, VirusTotal will allow us to know relevant details about the

[2]https://www.virustotal.com

downloaded binaries. Thus, an SSH session will be tagged as positive when:

- A change in honeypot status or an outgoing network connection occurs when the attacker logs out.
- The VirusTotal scanning indicates that the downloaded executable during that session corresponds to a malware.

Once the data was tagged, we trained and evaluated the data to produce three different models and assess if they can be used to classify new SSH sessions. We defined the models for executed commands, for network features, and a hybrid model using both executed commands and network features.

In the next sections, we explain into detail how the data was gathered, the metrics used, the model, as well as the technical features of the developed system.

5. Dataset, Metrics, and Model

A dataset is a table where each column represents a feature, and each row is a real, complete, and well-labeled sample of the dataset [42]. Rapid7 Heisenberg Cloud Honeypot[3] and CIC Honeynet[4] SSH datasets are the most commonly used datasets in SSH attack research. However, none of these datasets have SSH sessions' network traffic information to train our system model. Both datasets are based on Cowrie [43], which is a medium interaction SSH and Telnet honeypot designed to log brute force attacks and the shell interaction performed by the attacker.

Our dataset consists of executed commands and network traffic statistics generated during the SSH sessions. It represents what attackers perform during SSH sessions and whether SSH sessions have infected the honeypot or not. Session tagging has been done with the labeling strategy described in Section 4. Table 3 shows a sample record of a Dota attack [44] that our honeypots received.

All sessions used in our dataset were captured from different HIH deployed in Europe, Asia, the US, North America, and South America. Azure[5] virtual

[3]https://opendata.rapid7.com/heisenberg.cowrie/
[4]https://www.honeynetproject.com/dataset.html
[5]https://azure.microsoft.com/

Feature	Value
Session ID	0521d...36be7
Attacker's IP	87.64.253.40
Session duration	141 s
Executed commands	cat /proc/cpuinfo | grep name | wc -l echo "root:ECO3kGZ0lsqI"|chpasswd|bash ... rm -rf /var/tmp/dota* put /var/tmp/dota.tar.gz ... sleep 15s echo "IyE3L...0IDA=" | base64 –decode | bash
ips_src_count	1
ips_dst_count	1
urls	[]
sent_bytes	375802
received_bytes	6091334
tcp_bytes	6467136
udp_bytes	0
size_payloads	6072192
size_first_packet	146
size_last_packet	146
max_size_sent_packet	354
min_size_sent_packet	66
max_size_received_packet	4274
min_size_received_packet	66

Table 3: Record of the Dota attack.

hosts have been used, with Ubuntu Server 18.04 LTS, 2 virtual CPUs, and 4 GB of RAM. The HonSSH honeypots were configured to allow unlimited access via SSH service. Therefore, any credential used by the attackers was accepted. Also, we installed exploitable services in the honeypots, and we increased the physical characteristics of our honeypots to make them more attractive to attackers as well.

After five months, our dataset was made up of 20,759 SSH sessions, but

18,826 were sessions with no executed commands by the attacker, it seems that the attacker just wanted to check the access into the machine looking forward to a possible future attack. Thus, these sessions were removed from our dataset; otherwise, our model would have obtained results with very high success rates even though our system does not detect positive sessions correctly. As a result of the application of the sanitizing process, we obtained a dataset with 2139 records: 337 positive SSH sessions (infected host) and 1802 negative SSH sessions (uninfected host).

Next, we performed the processing of capture data for each SSH session to get our features. As shown in Table 3, an SSH session contains a list of strings (executed commands), which must be split into individual commands. We developed a parser to retrieve only the command itself, without parameters. Thus, we can trace executed commands during an SSH session, and where each command will represent an individual feature. Highlight that we selected only a pool of 72 commands from a list of the 100 most interesting Unix commands. Other 28 commands were deemed uninteresting to be added to our dataset because they were only used to improve the legitimate user experience.

So, this dataset has a total of 93 features: 72 commands, 7 session states, and 14 network statistics. Commands represent a list of the different commands executed in all attacks received. Session states refer to the number of executed commands in the ones defined by [26]: *Check software* configuration, *Install* a program, *Download* a file, *Run* a rogue program, *Change* the account *password*, *Check the hardware* configuration, and *Change the system* configuration. Network statistics are features such as session duration, bytes sent, bytes received, etc. Table 2 shows all the network features.

Three different models were defined according to the features used to train it: (1) model trained with the SSH commands executed by attackers, (2) model trained with the network statistics features, and (3) a hybrid model trained with executed commands and network features. The most widely used Supervised Learning algorithms in previous ML-based botnet detection research have been used to assess these models: Decision Tree, Random Forest, Support Vector Machine, and Naive Bayes. The ML algorithms used in this research are implemented by scikit-learn. The most relevant attack detection evaluation metrics were chosen for comparison between different models and algorithms [45], which are described in Section 2.3.

In our study, to split the dataset into training and testing, we met the following constraints:

1. The training and cross-validation dataset was approximately 80% of the dataset.
2. The testing dataset was 20% of the dataset.
3. None of the records used in the training and cross-validation [46] dataset was used in the testing dataset.

The dataset division was random and repeated a finite number of times to avoid different results depending on how the data were distributed in the subsets, always respecting the previous constraints. Herein, we wanted to find a simple machine learning model with enough classification performance.

Table 4 showcases the results obtained by the different classification algorithms proposed, where the considered metrics are evaluated by taking the sessions that are labeled "infected" as the target class. To reduce standard error, these results are the median of all tests conducted by each algorithm.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
Decision Tree	89.4	100	43.1	60.3
Random Forest	**98.1**	**95.7**	**93.9**	**94.8**
SVM	97.7	96.7	90.8	93.7
Naive Bayes	69.9	38.2	98.5	55.0

Table 4: Performance of the classification algorithms proposed.

The ML algorithms achieving the best results were SVM and RF. SVM obtained 96.7% of precision, indicating that between all cases classified as positive the 96.7% were positive, and 90.8% of recall, denoting that it was able to sort the 90.8% of all positive cases in the dataset correctly. On the other hand, RF obtained 95.7% of precision, having a higher number of false positives than SVM. Instead, RF obtained 93.9% of recall indicating that it got a lower number of false negatives. Both classification errors are not a valid approach as a result, but a smaller number of false negatives is preferable in security problems, intending to avoid the potential risks in case threats are not well detected. Although both models achieve similar results, we have chosen the best classification model generated by the RF classifier due to a lower number of false negatives and, mainly, because RF is computationally less expensive than SVM.

Finally, an exhaustive search of the parameters for the RF classifier [47] was performed. We aimed to find the most optimal parameters that obtain the best classification performance, without model overfitting. To do this,

we used K-fold Cross-Validation [48], which guarantees that results are independent of the split of the dataset. The set of hyperparameters and their respective ranges are listed in Table 5. The best results reached in our study, without model overfitting, were obtained by the configuration 1600-90-5-1-auto-True.

Hyper-parameter	Values tested
n_estimators	200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000
max_depth	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None
min_samples_split	2, 5, 10
min_samples_leaf	1, 2, 4
max_features	auto, sqrt
bootstrap	True, False

Table 5: Configurations and hyper-parameters for our test.

To summarize, our model was based on the Random Forest classifier. In addition to the high level of prediction achieved, this algorithm offers suitable characteristics for threat detection environments. Researchers are increasingly using Random Forest for predictive data modeling due to its high accuracy, time efficiency, and low computational cost [49], which makes this algorithm the best choice for hosts without vast capabilities that especially need a real-time response.

6. Experimental Results and Discussions

In the evaluation process, classification accuracy and other error metrics were used to show the effectiveness of our model, tested with a training dataset consisting of 427 examples (20% of the dataset): 54 positive SSH sessions (infected host) and 373 negative SSH sessions (uninfected host). In classification problems, classification errors (false positives and false negatives) are more critical than classification hits. It means that a model that detects all events as positive can have a high percentage of accuracy, but it is not a good model. The results of our model are shown in Table 6.

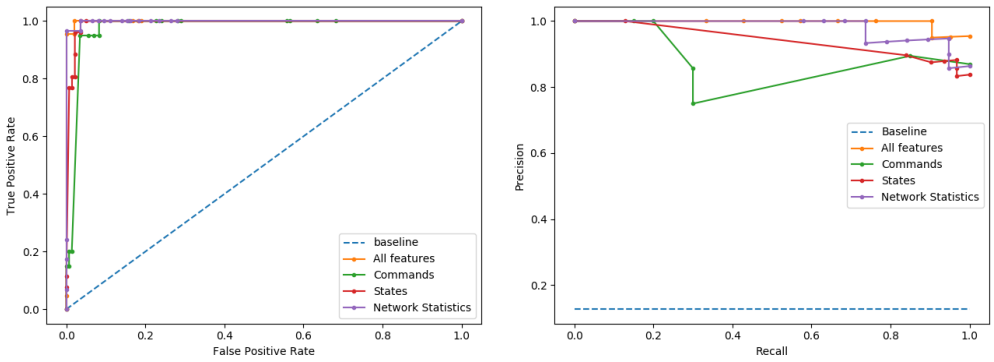
After the tests were performed, our system was able to predict SSH infection with very high accuracy (99.59%). Moreover, our model reached a precision of 96.87% and a recall of 100%, thus giving an F1 score of 98.41%. Not all uninfected SSH sessions were detected. However, all malicious SSH

Class	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
Infected	99.59	96.87	100	98.41
Uninfected	99.59	100	99.53	99.76

Table 6: Model performance for the training dataset.

sessions were correctly identified (false negatives = 0). Not having false negatives is very critical in an attack detection system.

Figure 2 shows ROC and Precision-Recall curves for models that have been trained with different features (all features, single commands, only network statistics, and only session states). We aim to show what information best represents an attacker's behavior for an infected SSH session. As we can see, the best result is reached by the model trained with all features. We want to highlight that the network statistics profile the attacker's behavior better than commands executed by attackers. Thus, it confirms our hypothesis that commands executed by the attacker can give us a lot of information about the purpose of that SSH session. Still, since several commands are performing the same operation, additional context information, such as network statistics, is needed to predict attacks accurately. Unfortunately, session states do not seem to represent the infection phase correctly via SSH service.



(a) ROC curve

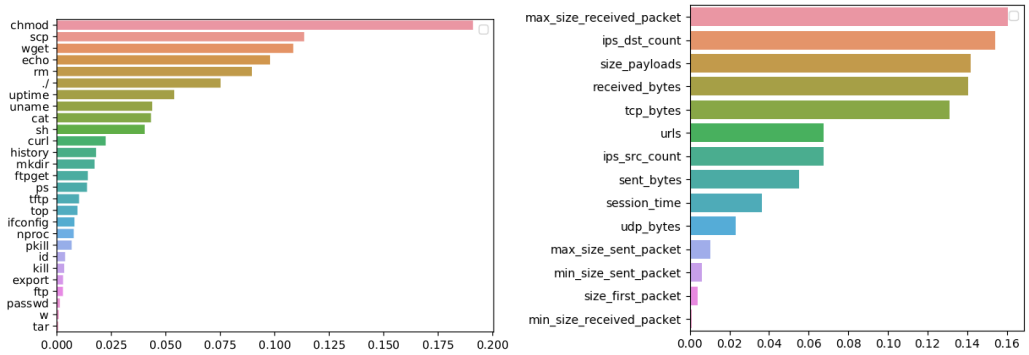
(b) Precision-Recall curve

Figure 2: ROC curves and Precision-Recall curves for classification.

The importance of each command in the detection process of our tests is shown in Figure 3a. The most relevant command is *chmod*, commonly used

to give execution permission to a file to execute it. *Scp* and *wget* allow the transfer and download a file into the victim. Other significant commands are *echo*, *rm*, and *./* (run an executable file). These commands correspond to *Download*, *Installation*, and *Execution* states, which matches a sequence widely used in this attack type [26]. These results do not mean that attackers cannot infect a host by executing other commands, but it is the most frequently seen pattern in the received attacks.

Concerning network statistics (in Figure 3b), the features that provide the most value to our model are: *max_size_received_packet*, *ips_dst_count*, *size_payloads*, *received_bytes*, *tcp_bytes*, and *urls*. These features can be the number and destination of connections that the attacker makes from the vulnerable host to the C&C server. The attacker uses this server to download malware (with a particular size) to turn his/her victim into a bot. These connections are prevalent in a typical bot infection phase [13].



(a) Commands (b) Network statistics

Figure 3: Features importance scores.

To finish with, we have obtained results with a very high-performance level. Our model presents an accuracy of 99.59% and can detect all SSH sessions intended to infect the honeypot. It should be noted that our work is focused on identifying a botnet's infection phase, especially infection via SSH service. As seen in Section 3, no related work was focused on this initial phase. Our achieved results match and even improve the previous work of botnet detection with machine learning.

Nevertheless, no real comparison can be made with previous works, since existing datasets with network traffic reflect phases other than that of infection and the commands executed by attackers only fit with our scenario.

Instead, we have evaluated our approach with a dataset created by our honeypots for an SSH attack environment. Thus, we can confirm that Machine Learning techniques are ideal for our detection system.

7. Conclusion and Future Work

The new malware and infected computers are continuously appearing every day. In recent years, a large number of servers and IoT devices have been infected via SSH service to perform DDoS attacks, among others. Furthermore, zero-day malware and variants succeed in evading the traditional methods. Thus, new botnet detection techniques are needed.

In this paper, we have presented a Machine Learning-based approach for the detection of incoming SSH security threats in real-time, aiming to infect new devices. Our main objective has been to detect the attacker before the targeted device gets infected, and thus prevent the victim host from participating in any malicious activity. To achieve this, we have generated a novel labeled dataset composed of executed commands and network features generated during an SSH session. For this purpose, high interaction SSH honeypots have been deployed to capture a botnet's infection phase using the SSH service. After the data capture period, we developed an infection detection model that reached a very high level of prediction and zero false negatives. Indeed, our system detected all known and unknown SSH sessions intended to infect our honeypots, and not having false negatives. Thus, our research has demonstrated that new SSH infections can be detected through Machine Learning techniques.

Future perspectives are currently being considered. Firstly, to train and evaluate the performance of the proposed model with a larger dataset. Moreover, the improvement of the feature selection process and adaptation of the proposed model to new threats through the analysis of new malware that spreads via the SSH service. Also, it would be interesting to integrate our system with a Black Hole Route, dropping all packets sent from a source IP detected by our model as a potentially malicious IP. Finally, we propose to adapt our approach for its use in a real environment, using a reverse SSH proxy to control SSH sessions between clients and real devices, and thus classify SSH sessions as infected or uninfected in near real-time.

References

[1] Spamhaus Malware Labs, Spamhaus botnet threat report 2019, https://www.spamhaus.org/news/article/793/spamhaus-botnet-threat-report-2019, Accessed 11 May 2020 (2019).

[2] C. Silverman, 8 people are facing charges as a result of the FBI's biggest-ever ad fraud investigation, https://www.buzzfeednews.com/article/craigsilverman/3ve-botnet-ad-fraud-fbi-takedown, Accessed 11 May 2020 (2018).

[3] K. Chen, S. Zhang, Z. Li, Y. Zhang, Q. Deng, S. Ray, Y. Jin, Internet-of-Things security and vulnerabilities: Taxonomy, challenges, and practice, Journal of Hardware and Systems Security 2 (2) (2018) 97–110. doi:10.1007/s41635-017-0029-7.

[4] C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, DDoS in the IoT: Mirai and other botnets, Computer 50 (7) (2017) 80–84. doi:10.1109/MC.2017.201.

[5] R. Millman, OVH suffers 1.1Tbps DDoS attack, https://www.scmagazineuk.com/ovh-suffers-11tbps-ddos-attack/article/1476220, Accessed 11 May 2020 (2016).

[6] T. Easton, Chalubo botnet wants to DDoS from your server or IoT device, https://news.sophos.com/en-us/2018/10/22/chalubo-botnet-wants-to-ddos-from-your-server-or-iot-device, Accessed 11 May 2020 (2018).

[7] Trend Micro, Outlaw hacking group's botnet observed spreading Miner, Perl-based backdoor, https://blog.trendmicro.com/trendlabs-security-intelligence/outlaw-hacking-groups-botnet-observed-spreading-miner-perl-based-backdoor/, Accessed 11 May 2020 (2019).

[8] T. Ylonen, C. Lonvick, IETF RFC 4254: The Secure Shell (SSH) connection protocol, https://tools.ietf.org/html/rfc4254, Accessed 11 May 2020 (2006).

[9] S. S. Silva, R. M. Silva, R. C. Pinto, R. M. Salles, Botnets: A survey, Computer Networks 57 (2) (2013) 378–403. doi:10.1016/j.comnet.2012.07.021.

[10] G. Vormayr, T. Zseby, J. Fabini, Botnet communication patterns, IEEE Communications Surveys & Tutorials 19 (4) (2017) 2768–2796. doi:10.1109/COMST.2017.2749442.

[11] P. Wainwright, H. Kettani, An analysis of botnet models, in: 2019 3rd International Conference on Compute and Data Analysis, 2019, pp. 116–121. doi:10.1145/3314545.3314562.

[12] S. K. Sahay, A. Sharma, H. Rathore, Evolution of malware and its detection techniques, in: Information and Communication Technology for Sustainable Development, 2020, pp. 139–150.

[13] M. Stevanovic, J. M. Pedersen, Machine learning for identifying botnet network traffic, Tech. rep., Department of Electronic Systems, Aalborg University (2013).

[14] S. Miller, C. Busby-Earle, The role of machine learning in botnet detection, in: 2016 11th International Conference for Internet Technology and Secured Transactions, 2016, pp. 359–364. doi:10.1109/ICITST.2016.7856730.

[15] Kaspersky, Honeypots and the Internet of Things, `https://securelist.com/honeypots-and-the-internet-of-things/78751/`, Accessed 11 May 2020 (2017).

[16] B. Çakir, Zero-day attack detection with deep learning, PhD Dissertation, Middle East Technical University, Turkey, `http://etd.lib.metu.edu.tr/upload/12624014/index.pdf`, Accessed 11 May 2020 (2019).

[17] M. Tsikerdekis, S. Zeadally, A. Schlesener, N. Sklavos, Approaches for preventing honeypot detection and compromise, in: 2018 Global Information Infrastructure and Networking Symposium, 2018, pp. 1–6. doi:10.1109/GIIS.2018.8635603.

[18] H. Choi, H. Lee, Identifying botnets by capturing group activities in DNS traffic, Computer Networks 56 (1) (2012) 20–33. doi:10.1016/j.comnet.2011.07.018.

[19] A. Pektaş, T. Acarman, Deep learning to detect botnet via network flow summaries, Neural Computing and Applications 31 (11) (2019) 8021–8033. doi:10.1007/s00521-018-3595-x.

[20] W. Wang, Y. Shang, Y. He, Y. Li, J. Liu, BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors, Information Sciences 511 (2020) 284–296. doi:10.1016/j.ins.2019.09.024.

[21] A. D. Biradar, B. Padmavathi, BotHook: A supervised machine learning approach for botnet detection using DNS query data, in: 2nd International Conference on Communications and Cyber Physical Engineering, 2020, pp. 261–269. doi:10.1007/978-981-13-8715-9 31.

[22] T. Bajtoš, P. Sokol, T. Mézešová, Virtual honeypots and detection of Telnet botnets, in: Central European Cybersecurity Conference, 2018, pp. 1–6. doi:10.1145/3277570.3277572.

[23] T. Bajtoš, P. Sokol, A. Gajdoš, K. Lučivjanská, T. Mézešová, Analysis of the infection and the injection phases of the Telnet botnets, Journal of Universal Computer Science 25 (11) (2019) 1417–1436. doi:10.3217/jucs-025-11-1417.

[24] E. C. Ogu, O. A. Ojesanmi, O. Awodele, S. Kuyoro, A botnets circumspection: The current threat landscape, and what we know so far, Information 10 (11) (2019) 337. doi:10.3390/info10110337.

[25] R. Marinho, R. Holanda, Exploring a P2P transient botnet - From discovery to enumeration, The Journal on Cybercrime & Digital Investigations 3 (1) (2017) 30–39. doi:10.18464/cybin.v3i1.16.

[26] D. Ramsbrock, R. Berthier, M. Cukier, Profiling attacker behavior following SSH compromises, in: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007, pp. 119–124. doi:10.1109/DSN.2007.76.

[27] G. K. Sadasivam, C. Hota, B. Anand, Towards extensible and adaptable methods in computing, Springer Singapore, 2018, Ch. Honeynet data analysis and distributed SSH brute-force attacks, pp. 107–118. doi:10.1007/978-981-13-2348-5 9.

[28] I. Koniaris, G. Papadimitriou, P. Nicopolitidis, Analysis and visualization of SSH attacks using honeypots, in: Eurocon 2013, 2013, pp. 65–72. doi:10.1109/EUROCON.2013.6624967.

[29] M. Banerjee, B. Agarwal, S. Samantaray, An integrated approach for botnet detection and prediction using honeynet and socialnet data, in: International Conference on Intelligent Computing and Smart Communication 2019, 2020, pp. 423–431. doi:10.1007/978-981-15-0633-8 41.

[30] D. Drinkwater, 5 top machine learning use cases for security, `https://www.csoonline.com/article/3240925/5-top-machine-learning-use-cases-for-security.html`, Accessed 11 May 2020 (2017).

[31] T. M. Mitchell, Machine learning, Burr Ridge, IL: McGraw Hill 45 (37) (1997) 870–877.

[32] K. Rieck, T. Holz, C. Willems, P. Düssel, P. Laskov, Learning and classification of malware behavior, in: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, 2008, pp. 108–125. doi:10.1007/978-3-540-70542-0 6.

[33] S. Nasiriany, G. Thomas, W. Wang, A. Yang, J. Listgarten, A. Sahai, A comprehensive guide to machine learning, CS 189 Guide, University of California, Berkeley, `http://snasiriany.me/cs189`, Accessed 11 May 2020 (2019).

[34] E. B. Beigi, H. H. Jazi, N. Stakhanova, A. A. Ghorbani, Towards effective feature selection in machine learning-based botnet detection approaches, in: 2014 IEEE Conference on Communications and Network Security, 2014, pp. 247–255. doi:10.1109/CNS.2014.6997492.

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

[36] S. Shin, Z. Xu, G. Gu, EFFORT: Efficient and effective bot malware detection, in: 2012 Proceedings IEEE INFOCOM, 2012, pp. 2846–2850. doi:10.1109/INFCOM.2012.6195713.

[37] M. Stevanovic, J. M. Pedersen, An efficient flow-based botnet detection using supervised machine learning, in: 2014 International Conference on Computing, Networking and Communications, 2014, pp. 797–801. doi:10.1109/ICCNC.2014.6785439.

[38] G. Kirubavathi, R. Anitha, Botnet detection via mining of traffic flow characteristics, Computers & Electrical Engineering 50 (2016) 91–101. doi:10.1016/j.compeleceng.2016.01.012.

[39] 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.

[40] R. U. Khan, R. Kumar, M. Alazab, X. Zhang, A hybrid technique to detect botnets, based on P2P traffic similarity, in: 2019 Cybersecurity and Cyberforensics Conference, 2019, pp. 136–142. doi:10.1109/CCC.2019.00008.

[41] T. Nicholson, HonSSH, `https://github.com/tnich/honssh`, Accessed 11 May 2020 (2018).

[42] 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:10.1016/j.cose.2011.12.012.

[43] Cowrie Project, Cowrie SSH/Telnet honeypot, `https://github.com/cowrie/cowrie`, Accessed 11 May 2020 (2020).

[44] Kindredsec, Dota Campaign: Analyzing a coin mining and remote access hybrid campaign, `https://kindredsec.com/2019/05/31/dota-campaign-analyzing-a-coin-mining-and-backdoor-malware-hybrid-campaign/`, Accessed 11 May 2020 (2019).

[45] L. Fernández Maimó, A. L. Perales Gómez, F. J. García Clemente, M. Gil Pérez, G. Martínez Pérez, A self-adaptive deep learning-based system for anomaly detection in 5G networks, IEEE Access 6 (2018) 7700–7712. doi:10.1109/ACCESS.2018.2803446.

[46] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: International Joint Conference on Artificial Intelligence, 1995, pp. 1137–1145.

[47] Scikit-learn, RandomForestClassifier, `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`, Accessed 11 May 2020 (2019).

[48] T. Fushiki, Estimation of prediction error by using K-fold cross-validation, Statistics and Computing 21 (2) (2011) 137–146. doi:10.1007/s11222-009-9153-8.

[49] K. Singh, S. C. Guntuku, A. Thakur, C. Hota, Big data analytics framework for peer-to-peer botnet detection using random forests, Information Sciences 278 (2014) 488–497. doi:10.1016/j.ins.2014.03.066.