Evaluation of Apache Spot's machine learning capabilities in an SDN/NFV enabled environment

Workshop paper

Christos M. Mathas
University of Peloponnese
Greece
mathas.ch.m@uop.gr

Olga E. Segou, PhD
Orion Innovations PC
Greece
osegou@orioninnovations.gr

Georgios Xylouris
Orion Innovations PC
Greece
gxilouris@orioninnovations.gr

Dimitris Christinakis
Orion Innovations PC
Greece
dchristinakis@orioninnovations.gr

Michail-Alexandros Kourtis
Institute of Informatics and Telecommunications
National Centre for Scientific Research "Demokritos"
Greece
akis.kourtis@iit.demokritos.gr

Costas Vassilakis
University of Peloponnese
Greece
costas@uop.gr

Anastasios Kourtis
Institute of Informatics and Telecommunications
National Centre for Scientific Research "Demokritos"
Greece
kourtis@iit.demokritos.gr

ABSTRACT

Software Defined Networking (SDN) and Network Function Virtualisation (NFV) are transforming modern networks towards a service-oriented architecture. At the same time, the cybersecurity industry is rapidly adopting Machine Learning (ML) algorithms to improve detection and mitigation of complex attacks. Traditional intrusion detection systems perform signature-based detection, based on well-known malicious traffic patterns that signify potential attacks. The main drawback of this method is that attack patterns need to be known in advance and signatures must be preconfigured. Hence, typical systems fail to detect a zero-day attack or an attack with unknown signature. This work considers the use of machine learning for advanced anomaly detection, and specifically deploys the Apache Spot ML framework on an SDN/NFV-enabled testbed running cybersecurity services as Virtual Network Functions (VNFs). VNFs are used to capture traffic for ingestion by the ML algorithm and apply mitigation measures in case of a detected anomaly. Apache Spot utilises Latent Dirichlet Allocation to identify anomalous traffic patterns in Netflow, DNS and proxy data. The overall performance of Apache Spot is evaluated by deploying Denial of Service (Slowloris, BoNeSi) and a Data Exfiltration attack (iodine).

ARES 2018, August 27–30, 2018, Hamburg, Germany

ACM ISBN 978-1-4503-6448-5/18/08…$15.00
https://doi.org/10.1145/3230833.3233278

CCS CONCEPTS

• **Computer systems organization** → **Distributed Architectures**; *Cloud Computing* • **Networks** → Network Security; Network Services • **Computing Methodologies** → Machine Learning.

KEYWORDS

Software Defined Networking, Network Function Virtualisation, Machine Learning, Latent Dirichlet Allocation, Apache Spot, Penetration Testing, SHIELD Project.

ACM Reference format:

C. M. Mathas, O. E. Segou, G. Xylouris, D. Christinakis, M. A. Kourtis, C. Vassilakis and A. Kourtis. 2018. SIG Proceedings Paper in word Format. In *Proceedings of CyberTIM Workshop, ARES conference, Hamburg, Germany, August 2018 (CyberTIM Workshop, ARES Conference 2018)*, 10 pages.

1 INTRODUCTION

The increasing demand for high-speed internet access has led to the evolution of the Internet towards softwarised and more scalable architectures. Software Defined Networking (SDN) [1] and Network Function Virtualisation (NFV) [2] are considered two key enabling technologies that underline the evolution of future infrastructures. The SDN/NFV paradigm enables ISPs to

compose and manage complex network services at the click of a button, while drastically improving time and cost efficiency [3].

At the same time, the growing number of complex cyber threats is often overlooked by traditional cybersecurity systems, due to the size, variety and velocity of traffic data that need to be inspected for signs of malicious activity. Cyber-attacks are observed with increasing frequency and impact radius, even on global scale [4]. With that in mind, governments and organizations are increasing their cybersecurity investment to ensure the integrity of their data and networks, focusing on advanced systems for detection and mitigation. Signature-based intrusion detection systems [5] search for known suspicious patterns that are often attributed to known cyber-attacks. The obvious drawback is that the attack signatures need to be known in advance and the detection techniques are too rigid, making detection harder in case of a multi-vector attack or zero-day exploit. Anomaly detection systems [6], on the other hand, can be trained with typical network traffic to create a baseline of normal traffic. This is especially critical when there is no attack signature associated with the malicious activity. Misclassified results, such as false positive or false negative detection, however, can have adverse operational effects.

Previous research in this area has considered a variety of different ML algorithms for anomaly detection. The authors in [7] follow a more general approach and present research related to anomaly detection, without focusing on its practical use in network security. The work in [8] and [9] focuses on surveying intrusion detection in the context of cyber security through data mining and machine learning methods. A multitude of methods have been used for the implementation of machine learning classifiers in cyber security research for anomaly detection. One approach is that of Artificial Neural Networks (ANN) [10] that requires a more time-consuming preprocessing stage, an ANN training stage and an ANN decision stage. Another popular approach is the use of a Bayesian network [11]; in that work, Bayesian networks are used to classify operating system calls which are the result of the reception of TCP/IP packets. The EXPOSURE system [12] utilizes decision trees in order to perform passive DNS analysis. Other classifiers, like Random Forest [13] are combinations of decision trees and ensemble learning, while Naïve Bayes [14], approaches offer a simplistic implementation of a Bayes classifier.

This work considers the case of Apache Spot [15], a machine learning (ML)-based platform for anomaly detection, that utilises Latent Dirichlet Allocation [16] to detect unusual traffic patterns. Latent Dirichlet Allocation is a Natural Language Processing algorithm, which is a factor that sets it apart from the common machine learning classifiers in use. Natural Language Processing methods are easy to apply on the variety of different network traffic logs and improve the overall threat intelligence capabilities by including more sources of structured, human-readable, textual data.

Other systems offer similar capabilities, such as Sqrrl [17] and Apache Metron [18]; Apache Spot sets itself apart not only by its machine learning capabilities, but also from its Open Data Model (ODM). As machine learning becomes a mainstream technology that is present in many consumer products, the cybersecurity industry has been quick to adopt it to improve on the existing defense capabilities. ODM brings together all security-related data (event, user, network, endpoint, etc.) into a singular view that can be used to detect threats more effectively. It also provides the ability to share and reuse threat detection models, analytics, and more. This improves interoperability among anomaly detection platforms and fosters the creation of an open data community.

Apache Spot is deployed and evaluated in the SHIELD [19] [20] SDN/NFV-enabled testbed in Athens, Greece. SHIELD proposes "a universal solution for dynamically establishing and deploying virtual security infrastructures into ISP and corporate networks". The project builds on the huge momentum of Network Functions Virtualisation (NFV) in order to virtualise security appliances into virtual Network Security Functions (vNSFs), to be instantiated within the network infrastructure.

Spot's integration with SDN/NFV leads to improved data ingestion, while cybersecurity VNFs can immediately receive threat information and apply mitigation measures. Its overall performance is tested using various penetration testing tools. In this work UDP flooding [21], Slowloris [21] and DNS Tunneling [22] attacks are performed. These attacks were selected due to the variation in the attack traffic in terms of velocity, variety and volume. In the case of UDP flooding, the attack traffic features high volume and velocity. Slowloris is a protocol-based Denial of Service attack that generates much smaller amounts of malicious incoming traffic. DNS tunneling is used as an example of a DNS-based data exfiltration attack that generates small amounts of traffic.

The rest of this paper is organized as follows: Section 2 presents the SDN/NFV testbed and Spot's core functions, deployment and mode of operation. It also presents the penetration testing tools used to perform the attacks against the network. Section 3 analyses the results of the penetration tests. Section 4 concludes this paper and discusses future work. The presented analysis aims to explore the performance of Apache Spot in this network setup, evaluate machine learning-based attack detection, and investigate Apache Spot configuration optimization for improvement of its performance in an SDN/NFV context.

2 EXPERIMENTAL AND COMPUTATIONAL DETAILS

2.1 The SHIELD Environment

SHIELD aims to develop a next-generation cybersecurity platform tailored for software networks, based on the SDN/NFV paradigm, big data and infrastructure attestation. The core components of the SHIELD platform (Figure 1) are deployed in two SDN/NFV testbeds in Athens and Barcelona, and include [23] [24]:

- **The vNSF Orchestrator (vNSFO):** vNSFO is based on the ETSI-supported open-source Management and Orchestration framework (Open Source MANO). The OSM orchestrator [25] is used to manage the lifecycle of the vNSFs. OSM on-boards vNSF packages, instantiates Network Services (NSs) in specific points of presence within the network infrastructure and monitors the running services.
- **The Virtual Infrastructure Manager (VIM):** ETSI-MANO [26] defines the VIM as the framework that manages computing, storage and network resources. SHIELD deploys the OpenStack platform as a VIM.
- **The SDN Controller:** OpenDaylight Carbon [27] is used as the SDN controller.
- **The Data Analysis and Remediation Engine (DARE)**: The DARE is an information-driven intrusion detection and prevention platform that stores and analyses heterogeneous network information, previously collected via the vNSFs.
- **The virtual Network Security Functions (vNSFs)**: OSM deploys cybersecurity as-a-Service in the form of KVM-based vNSFs (such as firewalls, deep packet inspection, intrusion detection etc.)
- **The vNSF Store:** Registers valid vNSF images from vNSF developers.
- **The Trust Monitor** [28]**:** Integrity of the NFVI, the vNSFs and the SDN Controller is checked periodically to detect compromised software and/or hardware. This work is based on the Trusted Computing paradigm and its Remote Attestation workflow.
- **The Security dashboard and controller**: The dashboard provides an overview of the security status and allows the operators to apply remediation actions received by the remediation engine of the DARE.

The DARE features cognitive and analytical components capable of predicting specific vulnerabilities and attacks. The processing and analysis of large amounts of data is carried out using big data analytics and ML techniques. Furthermore, the DARE remediation engine utilises the outputs from the cognitive and data analytics modules and various contextual information to determine a mitigation plan for the existing threats. The mitigation actions are relayed to the vNSFO in the XML-based Medium-Level Security Policy Language (MSPL) format [29]. The vNSFO then sends the appropriate rules to the vNSF, which applies them. For example, SHIELD deploys an OVS-based firewall vNSFs in a KVM/CentOS virtual machine, which receives rules to block IP addresses etc. This paper focuses on Apache Spot which is deployed as a DARE component. The anomalies detected by Apache Spot are passed to the remediation engine, which provides specific rules to the SHIELD vNSFs, such as blocking or rate limiting rules etc. This paper focuses on the Athens experimentation testbed, which is deployed and maintained by the National Centre for Scientific Research "Demokritos" (NCSRD), Orion Innovations P.C. and Space Hellas S.A. The Athens testbed provides ETSI-compliant [2] Network Function Virtualisation Infrastructure (NFVI) and deploys vNSFs, such as virtualized Firewalls, Deep Packet Inspection, Intrusion Detection and an instance of Apache Spot.

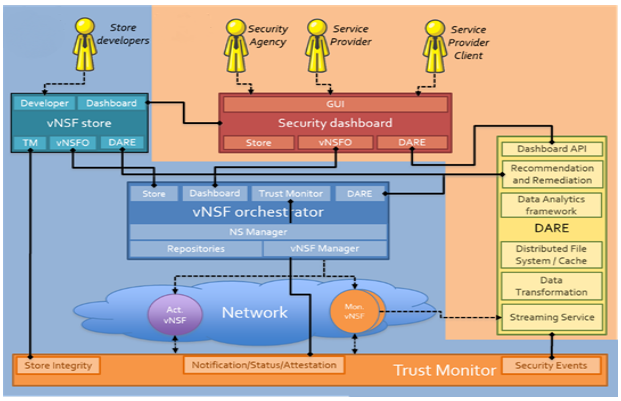

Figure 1: The SHIELD Platform.

2.2 Apache Spot

Apache Spot is an open-source platform that aims to facilitate the detection of cyber threats by using machine learning, rather than traditional signature-based detection. Apache Spot collects NetFlow, DNS and proxy data and analyses them to learn what constitutes typical network traffic. Anomalous traffic patterns are then identified and reported. The following subsections present Spot's core functions (Figure 2).

2.2.1 Data Ingestion

The data ingestion framework follows a distributed architecture that minimises the possibility of data loss and

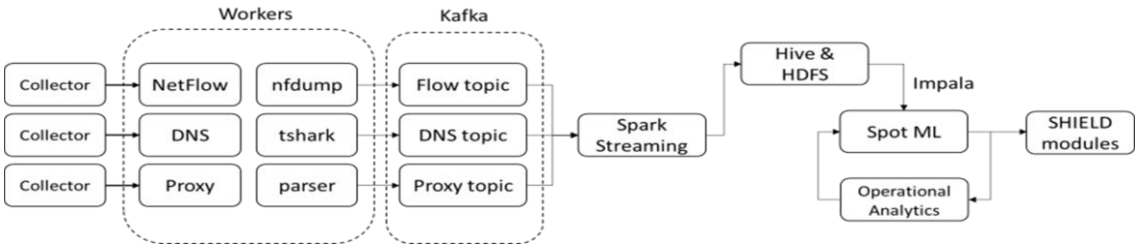

Figure 2: Apache Spot core functions.

ensures the availability of the service, even under heavy data loads that require significant processing power.

NetFlow, DNS and proxy data are captured in the network in nfcapd[1], pcap[2] and bluecoat[3] formats, respectively. The corresponding files are aggregated in Spot's filesystem, which is monitored by Collector daemons. Spot's Collectors initiate the ingestion process when new data are found in the filesystem. At this point, the data are human-readable. After the collection of the data, Spot's Worker daemons ingest the data, convert them to Avro-Parquet format[4] and place them in a Hive database. Workers are connected to Apache Kafka topics and partitions. There are two kinds of workers:

- Python workers which use multithreading to process the NetFlow and DNS data, and
- Spark streaming workers which execute a Spark application to read data from Apache Kafka.

The ingestion of the data in Hive tables at this point is necessary, since it prepares them in a form that is easily parsed by the Machine Learning algorithm.

2.2.2 Machine Learning

Apache Spot's machine learning includes routines for analyzing suspicious connects from the collection of the previously ingested data, based on Latent Dirichlet Allocation (LDA) [16]. LDA is a generative probabilistic model for collections of discrete data such as text corpora. Specifically, LDA is a topic modeling algorithm that defines a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. For its usage in Apache Spot it has been implemented using Spark MLlib [30]. Apache Spot applies LDA to network traffic, by converting the log entries to words through aggregation and discretization. This way, documents correspond to IP addresses, words to log entries (corresponding to an IP address) and topics to normal network traffic profiles. As a result, Apache Spot deduces a probabilistic model for each IP's behavior. The model assigns to each log entry a probability. The events (log entries) with the lowest probabilities are flagged as suspicious for further analysis. The output of this analysis is a list which includes the events that are less probable to occur according to a model of normal traffic conditions. Less probability of encountering a specific type of traffic means that this traffic is considered by Spot as the most suspicious.

2.2.3 Operational Analytics & Semi-Supervised Learning

Apache Spot provides a graphical Operational Analytics (OA) interface where it analyzes and presents the results of the machine learning algorithm. The results are divided in NetFlow, DNS, Proxy and by date of creation. Apache Spot allows semi-supervised learning through its OA module, in the sense that the user can evaluate and affect the model by scoring IP addresses and ports in terms of possible risks. These scores become available for use in the next execution of the ML algorithm. In this experimental deployment, no scoring was performed on the user-side.

2.3 Penetration Testing

The following attacks were utilized to assess Spot's performance: Slowloris, UDP Flooding and DNS Tunneling.

2.3.1 Slowloris

Slowloris [31] is a protocol-based DoS attack, that exploits HTTP protocol mechanisms to overwhelm the target, without the need to generate a massive flood of traffic. This attack establishes a lot of partial HTTP connections, that are not allowed to complete or to time-out. It is, therefore, a low bandwidth attack with almost no side-effects on other services and ports. By sending partial requests, Slowloris establishes connections with the target server which remain active as the server waits for the requests to complete, thus depleting its network resources. The server quickly reaches its maximum number of concurrent connections, which results in Denial of Service. The fact that this attack uses legitimate and not malformed packets, low bandwidth, and doesn't affect other services makes it very difficult for classic intrusion detection systems to detect.

2.3.2 UDP Flooding

BoNeSi [32] is a tool for simulating DDoS attacks. The attack is deployed using the default settings, which results to the creation of a UDP flood. The UDP protocol is connectionless and sessionless, which means that it establishes connections without the use of a three-way handshake like the TCP protocol. This makes it easy to abuse in order to overwhelm the target with a flood of UDP packets containing random data and it also means that it doesn't require a lot of resources to be committed by the attacker, other than network bandwidth, to be successful. The target server tries to find the service to which these packets are addressed to by reading the data and when it fails to find the service, it replies with "ICMP destination unreachable" packets. The UDP flood differs from Slowloris in the sense that it requires massive amounts of traffic to compromise the target server.

2.3.3 DNS Tunneling

The DNS tunneling [22] attack is used to bypass firewall restrictions and exfiltrate data from a network and/or establish a command and control channel between a commanding host outside the firewalled network perimeter and a compromised/accomplice host inside this perimeter. The data or the commands and their responses, are encoded in DNS queries and directed to a DNS server operated by the attacker. The malicious server is assigned a domain, so that any DNS query containing this domain is forwarded to the attacker. Iodine [33] is a tool for sending IPV4 data through a DNS server, by deploying a DNS tunnel among two endpoints. The iodine client serves as the compromised system that sends data through the

[1] https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html
[2] https://en.wikipedia.org/wiki/Pcap
[3] ELFF-compatible format, http://www.w3.org/TR/WD-logfile.html
[4] https://avro.apache.org/docs/1.2.0/ and http://parquet.apache.org/

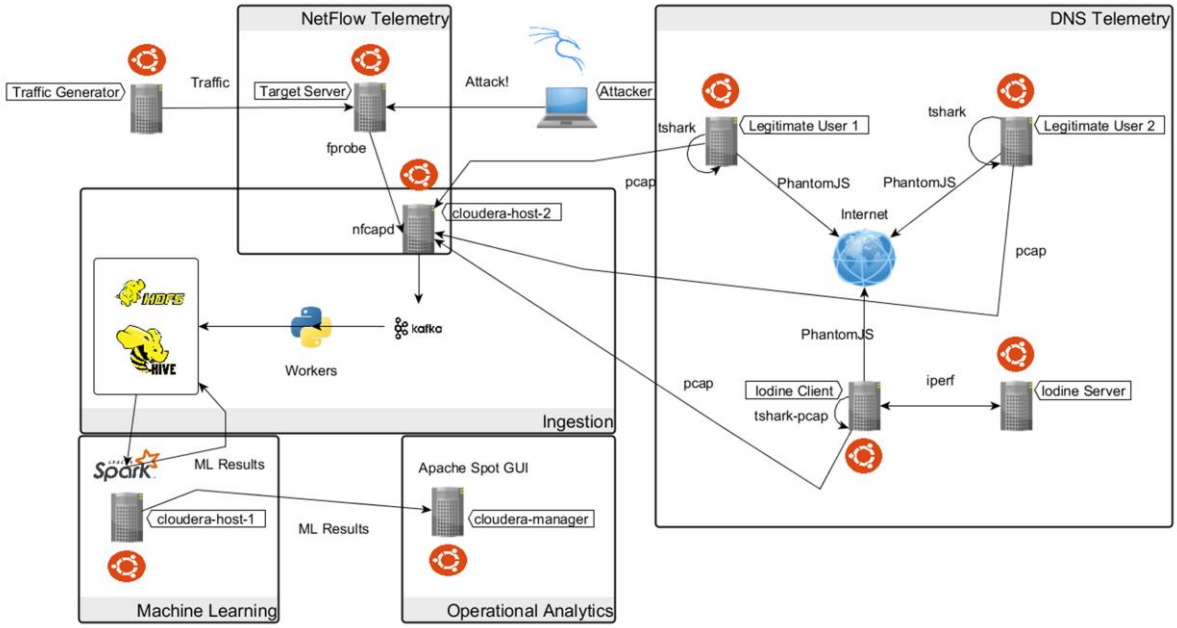

Figure 3: Deployment configuration of the experimental setup.

tunnel, and the iodine server is deployed as the malicious authoritative DNS server that receives the exfiltrated data or handles the command and control channel.

2.4 Deployment Configuration

The installation of Apache Spot (Figure 3) was made on a Cloudera Cluster version 5.11.1y. A minimum of three Cloudera nodes is required in order to deploy the Setup, Ingestion, Machine Learning and Operational Analytics components of Apache Spot. Three hosts were set up running Ubuntu Trusty 14.04 and the prerequisite components (HDFS, Hive, Impala, Kafka, Spark, Yarn, and Zookeeper). Our setup with the 3 Virtual Machines (VMs) was the following:

- *Cloudera-host-1.shield.com*: Configured as the Machine Learning (ML) Node, includes the Yarn Node Manager;
- *Cloudera-host-2.shield.com*: Configured as the Setup and Ingest Node, also known as the Edge Gateway (GW node), includes Kafka Broker and HDFS NFS Gateway; and
- *Cloudera-manager.shield.com*: Operational Analytics (OA) node, includes Cloudera Manager.

In addition to Spot, two virtual machines running Ubuntu 16.04 Server and a virtual machine running Kali Linux 2017.2 were deployed and configured as follows:

- The target server: The victim server that is under Apache Spot's supervision.
- The traffic generator: Replays files which contain normal, legitimate traffic.
- The attacker: A Kali Linux VM, which performs the attacks against the victim webserver.

Network traffic was recorded and anonymised from the network of the Media Networks Laboratory (at the Institute of Informatics and Telecommunications of NCSR Demokritos) and saved in pcap files. For the NetFlow Telemetry, the target server forwards the traffic from its network interface, using the *fprobe* tool, to *cloudera-host-2*, which is the server responsible for the data ingestion function of Apache Spot. Cloudera-host-2 records the traffic in nfcapd files. The attacks are deployed through Kali Linux in real time (during the replay of the network traffic from the traffic generator). This way, they are also forwarded to cloudera-host-2 and saved in the aforementioned nfcapd files.

For the DNS telemetry, three virtual machines running Ubuntu 16.04 Server and one virtual machine running Ubuntu 14.04 Server were deployed and configured as follows:

- The legitimate users 1 and 2: These users load well-known webpages and save the traffic in pcap files, thus creating normal DNS traffic.
- Iodine Client: Loads well-known websites, but additionally sends data using the *iperf* tool through a DNS tunnel which is established with the Iodine Server and these are saved in pcap files.
- Iodine Server: Receives the data sent by the Iodine Client using the *iperf* tool at the network interface of the DNS tunnel.

In the DNS tunneling scenario, PhantomJS [34] is used to create traffic towards Alexa's top 100 websites [35]. This generates the normal DNS traffic for this experiment, which is saved in pcap files ingested by Apache Spot. The iodine client signifies the compromised system that has set up the tunnel. Iperf [36] traffic is sent through the tunnel, encoded in the DNS

queries towards the malicious authoritative DNS server (the iodine server).

Each attack was executed ten times, and the results were collected and analyzed. The analysis results are presented and discussed in the following section.

3 RESULTS AND DISCUSSION

Apache Spot outputs the results of the ML algorithm in the form of a csv file. The csv file contains a ranking of the network traffic with the first entry being the one considered the most dangerous. Each entry contains the probability of being normal as calculated by the machine learning algorithm. Furthermore, each entry provides extensive information like date and time, source and destination IP addresses, source and destination ports, network protocol etc. The file is further processed to obtain the following metrics:

- The attack ranking, meaning the position of the first entry which is part of the attack. This metric is labeled as "Position" in our results.
- The probability assigned to the first entry which is part of the attack. This metric is labeled as "Probability" in our results.
- The number of entries which are part of the attack in the first 100 entries of the CSV file. This metric is labeled as "First 100" in our results.
- The percentage of the entries which are part of the attack for the whole CSV file. This metric is labeled as "Occurrence %" in our results.

In order to get a better visualization of the distribution between the ten attacks executed with each penetration testing tool and determine the performance of the classifier, results are presented in boxplot and Receiver Operating Characteristic (ROC) curve. A receiver operating characteristic (ROC) curve [37] is a technique used for visualizing the performance of binary classifiers. The plot is created by using the True Positive Rate of the classifier, also known as sensitivity and the False Positive Rate, also known as Fall-Out. For example, if a given result x is a true positive the curve moves 1/(#-of-positives) upwards and if false positive 1/(#-of-negatives) to the right. The area under the ROC curve, called AUROC, represents the probability that the classifier will rank a randomly chosen true positive instance higher than a randomly chosen false positive instance.

The boxplot (also known as a *box-and-whiskers plot*) is a single dimensional figure that illustrates the overall properties of a selected dataset in a simple visualization. The lower boundary of the box represents the first quartile Q1, i.e. the boundary that denotes 25% of the measurements. The median (Q2) lies within the box, and the top boundary of the box represents the Q3 quartile (75% of the measurements). The relative position of the median, with respect to Q1 and Q3, can indicate whether a distribution is symmetrical or skewed. The interquartile range IQR is defined as:

$$IQR = Q3 - Q1 \quad (1)$$

The system's range of normal operations is defined by the whiskers, which are calculated as a function of the interquartile range:

$$Lower\ whisker = Q1 - k * IQR \quad (2)$$

$$Upper\ whisker = Q3 + k * IQR \quad (3)$$

Where k is a constant, set to k=1.5 as per Tukey's recommendation [38]. If the lower and upper whisker exceed the range of the minimum and maximum value, they are set to match them. If there are measurements that lie outside of the range defined by the whiskers, they are considered outliers (Figure 4).

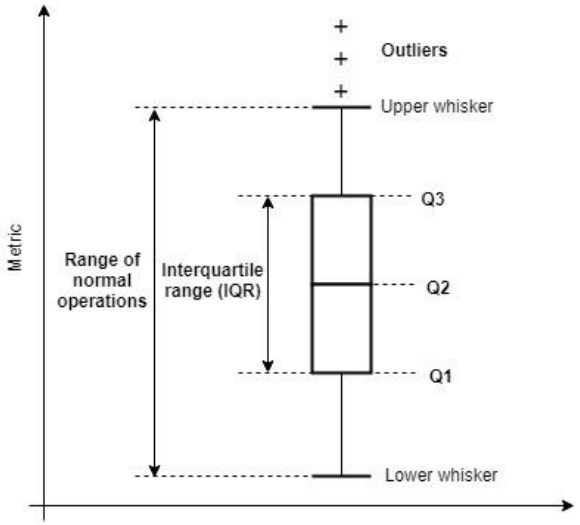

Figure 4: Box Plot Definition.

3.1 Slowloris

Table 1 illustrates experimental results after the ten executions of the Slowloris attack (in chronological order). The continuously open TCP connections make it stand out as an anomaly inside the data gathered from the network and thus it's being ranked as an anomaly. Although the first instance of the detected anomaly is ranked highly in terms of its position, the low number of anomalies in the first 100 results indicates the presence of numerous false positives. Slowloris is a low bandwidth attack taking up 2-4% of total traffic, as indicated by the low Occurrence percentage. A better indication of how efficiently Apache Spot detects the Slowloris attack is the high average AUROC value. AUROC evaluates the results of the machine learning model across the entire set of measurements.

Table 1: Slowloris test results.

Position	First 100	Probability	Occurrence %	AUROC
1	1	3.03E-05	3.388494878	0.70613
45	2	4.09E-05	3.109037018	0.83707
3	4	2.95E-05	3.802653298	0.71053
46	1	3.1E-05	2.958544408	0.80984
2	4	2.31E-05	2.6737318	0.78203
1	4	1.95E-05	2.842614412	0.76789
1	2	1.93E-05	2.782220821	0.75087
3	6	2.38E-05	2.995452056	0.79304
6	1	2.48E-05	2.626526191	0.82088
1	3	1.95E-05	2.647144165	0.82342
	Average	2.62E-05	2.982641905	0.780171
	Variance	4.67E-11	0.13871497	0.002128

3.2 BoNeSi

Table 2 depicts the results of ten executions of the BoNeSi attack (in chronological order). The BoNeSi attack was not successfully detected, which is attributed to the high occurrence percentage. The traffic generated by the UDP flood is much higher than normal traffic and thus disrupts the trained model. The algorithm starts considering the attack traffic as normal traffic rather than an anomaly. This is also supported by the low average AUROC value, which indicates that the BoNeSi traffic was considered as normal by the machine learning model.

Table 2: BoNeSi test results.

Position	First 100	Probability	Occurrence %	AUROC
40	2	3E-05	76.27055362	0.04333
36	3	1.98E-05	76.12198357	0.04168
63	2	3.92E-05	72.17966475	0.03832
28	2	2.08E-05	79.11053165	0.05949
124	0	5.88E-05	72.08022243	0.06492
132	0	0.000118	61.87038159	0.05606
47	2	3E-05	76.16703952	0.04501
72	3	3.92E-05	81.13165516	0.06449
29	3	1.31E-05	78.40206661	0.03994
105	0	3.15E-05	78.17122694	0.0405
	Average	4E-05	75.15053259	0.049373
	Variance	9.05E-10	29.88880099	0.000114

3.3 Comparison between Slowloris and BoNeSi

Figure 5 presents the box plots for the *Position* metric for both Slowloris and BoNeSi. The Slowloris box plot shows good results for the 75% of the distribution, but the remaining 25% reaches high values, thus the distribution is skewed. The BoNeSi box plot has a larger interquartile range showing that the results for the *Position* metric exhibit more variance in the case of BoNeSi. Figure 6 presents the box plots for the *Probability* metric for both Slowloris and BoNeSi. Similar values are observed up to the distribution's 75% percentile, although in the case for BoNeSi the top 25% measurements show significantly worse detection performance. The upper whisker in this case represents a useful metric and can be considered to be the threshold value for detection. Apache Spot allows the user to set a threshold value for anomaly detection and restrict the data that are analysed and visualized in its Operational Analytics. Considering that the overload of information that typically occurs in cyber security response, this is an important metric to estimate for a variety of cyber attacks. However, it is also shown that attacks with differing velocity and volume characteristics can vary in their detection thresholds. Figure 7 presents the ROC curve of the eighth Slowloris test as an example. Figure 8 illustrates the ROC curve of the second BoNeSi test as an example.

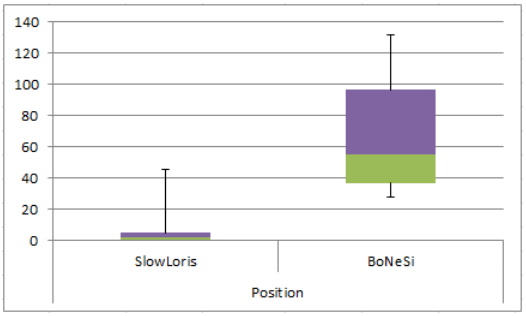

Figure 5: Position metric for Slowloris and BoNeSi.

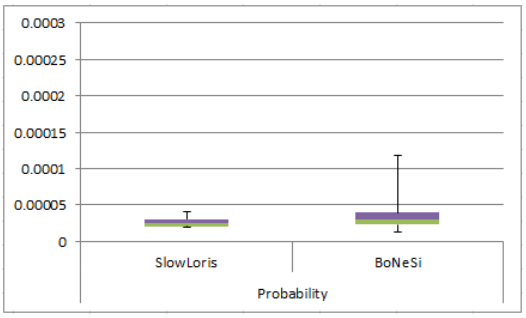

Figure 6: Probability metric for Slowloris and BoNeSi.

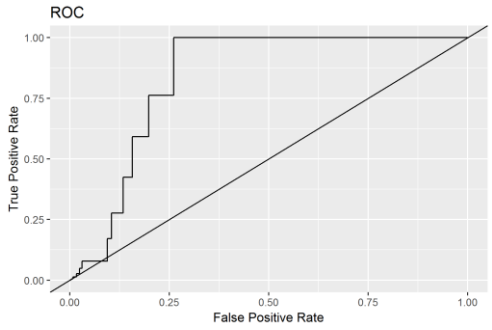

Figure 7: A ROC curve for Slowloris.

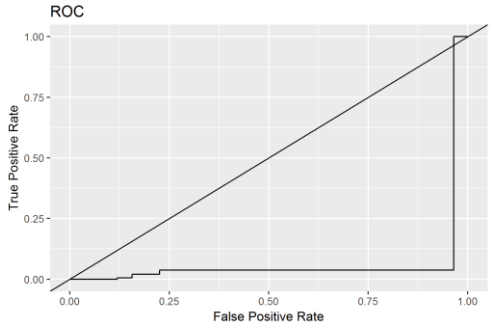

Figure 8: A ROC curve for BoNeSi.

3.4 Iodine

Table 3 presents the results of the ten executions of the Iodine attack experiment, while Figure 9 illustrates the ROC curve of the second test as an example. In the case of iodine, the results vary, showing that Spot does not effectively detect this attack. The low AUROC value shows that the classifier would need additional intelligence to better distinguish between false and true positives. Metrics such as the length of the DNS packet or the information entropy for the tunneled traffic would be necessary to improve the detection capabilities of Apache Spot in the context of this attack.

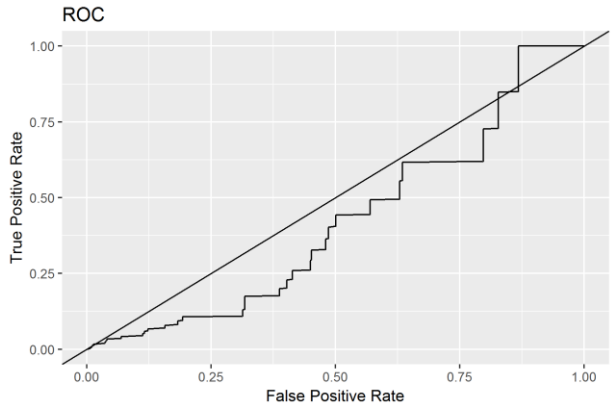

Figure 9: ROC curve of Iodine test #4.

Table 3: Iodine test results.

Position	First 100	Probability	Occurrence %	AUROC
4396	0	0.001214	2.654931102	0.106351
4832	0	3.72E-04	0.96459498	0.143839
1323	0	6.80E-04	3.266025207	0.141951
48	5	1.95E-04	4.329268293	0.421774
6599	0	3.50E-04	0.887902331	0.278647
81	4	3.82E-04	2.47187685	0.317081
619	0	0.001579	3.081664099	0.203787
3	8	2.35E-04	3.437132785	0.337798
1005	0	0.001846	3.58829084	0.121898
2	1	0.001191	6.375033342	0.138234
	Average	0.00080443	3.105671983	0.221136
	Variance	3.65E-07	2.518062774	0.012091

3.5 Varying traffic conditions

In order to assess how traffic volume affects the detection capabilities of Apache Spot, additional tests were performed with Slowloris, with a three-fold increase in normal traffic (in terms of number of flows) and a shorter attack window that produced less malicious flows. In effect, the occurrence percentage of the attack was decreased, and more normal traffic was provided to the machine learning model.

In Table 4 and Figure 10, the ML algorithm ranked about 90% of the results with even lower probability, as it stood out more as an anomaly. The first 100 worst ranked results still contained a large number of false positives just as in the case of the initial setup. This increase can be explained by the lower occurrence of the attack, which makes the attack stand up as an anomaly. Another factor contributing to the increase of AUROC is the lower number of attack flows which results in an increase of the true positive rate (ROC). Figure 11 compares the probabilities assigned to normal legitimate traffic versus the Slowloris attack traffic. Normal traffic is shown to feature a wider range of operations (between the whiskers) which is expected, as it is ranked as more probable. The area in the boxplot near the normal traffic lower whisker, shows that normal and attack traffic probabilities overlap. This designates the area where the false positives may occur. Additional computation or an appropriate classification algorithm is necessary in this case, to reduce the number of false positive detections. Hence, while Apache Spot can effectively detect anomalies in the network traffic, it is not able to distinguish between atypical legitimate traffic and attack traffic without further effort.

Table 4: Slowloris results for the modified traffic flow pattern.

Position	First 100	Probability	Occurrence %	AUROC
3	2	7.12E-06	0.752554059	0.92927
62	4	8.25E-06	0.436317896	0.97269
3	1	6.78E-06	0.407834793	0.97041
4	2	7.22E-06	0.512330365	0.97414
3	5	6.74E-06	0.668097628	0.96869
3	3	7.44E-06	0.578707049	0.97531
2	6	5.06E-06	0.788402504	0.93475
132	0	1.50E-05	0.754733042	0.94012
66	3	7.38E-06	0.584756899	0.95739
3	4	5.32E-06	0.710545745	0.96549
	Average	7.63E-06	0.619427998	0.958827
	Variance	7.64E-12	0.01865386	0.000309

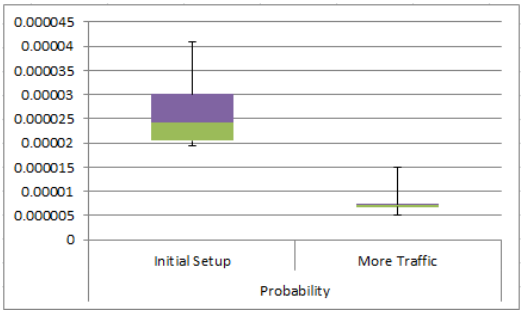

Figure 10: Probability metric for the Slowloris attack, initial vs. modified traffic flow pattern.

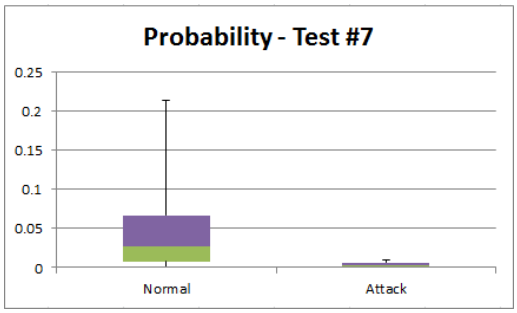

Figure 11: Probability metric for normal and attack traffic for the Slowloris attack, modified traffic flow pattern.

4 CONCLUSIONS

In this work, the Apache Spot machine learning framework was deployed and assessed in terms of its performance regarding the detection of three different attacks, namely Slowloris, UDP flooding and DNS tunneling. Results show that Apache Spot can successfully detect attacks such as Slowloris, which generate small amounts of network traffic and usually evade common intrusion detection systems (IDS). The results were less promising in the case of UDP flooding and DNS tunneling. In the case of UDP flooding, the amount of attack traffic disrupted the trained model and caused many misclassifications.

Results showed that the velocity and volume of the attack traffic, with respect to the normal traffic on the network, are determining factors that affect its detection capabilities. The machine learning algorithm requires a large amount of normal traffic to be trained and create a reference model for normal traffic. As normal traffic increases, true positive rates increase as well. False positives occurred in all analyses more frequently than false negatives, although they were ranked with low probabilities, as top threats. The high false positive rate may indicate a vulnerability that can lead to the exploitation of ML-based systems to induce a Denial-of-Service by misclassifying legitimate traffic as anomalous and applying remediation measures against it.

The results were positively affected by the increase in normal traffic and the decrease of the attack traffic which lead to a lower occurrence of the attack. Based on these conclusions, our future work is steered towards estimating how the ML algorithms would be vulnerable to poisoning attacks that would cause the training model to be compromised and cause misclassifications, in the form of false positives and false negatives.

ACKNOWLEDGMENTS

This work was partially supported by the SHIELD Project ("SHIELD: Securing against intruders and other threats through a NFV-enabled environment") funded under the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement No. 700199. This work was performed in the context of the diploma thesis: Christos M. Mathas, "Evaluation of Apache Spot by using penetration testing techniques" (Supervisor: Costas Vassilakis, 2018, original text in Greek) that was completed in the Department of Informatics and Telecommunications of the University of Peloponnese (Tripolis, Greece) and the Media Networks Laboratory, National Centre for Scientific Research "Demokritos" (Athens, Greece), accessible at https://soda.dit.uop.gr/sites/soda.dit.uop.gr/files/mathas-bsc-thesis-final.pdf .

REFERENCES

[1] H Kim and N Feamster, 2013, "Improving network management with software defined networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114-119.

[2] ETSI, 2014, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture," ETSI GS NFV-SWA 001 V1.1.1 (2014-12).

[3] BusinessWire, 2013, *Research and Markets: The Network Functions Virtualization (NFV) Market: Business Case, Market Analysis & Forecasts 2014 - 2019.*

[4] Symantec Security Center, 2018, "Internet Security Threat Report,".

[5] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung, 2013, "Intrusion detection system: A comprehensive review," *Elsevier Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16-24.

[6] P.García-Teodoro, J.Díaz-Verdejo, G.Maciá-Fernándeza, and E.Vázquez, 2009, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Elsevier Computers & Security*, vol. 28, no. 1-2, pp. 18-28.

[7] Arindam Banerjee, Vipin Kumar Varun Chandola, 2009 "Anomaly Detection:

A Survey," *ACM Computing Surveys*, vol. 41, no. 3, July 2009.

[8] D. K. Bhattacharyya, J. K. Kalita Monowar H. Bhuyan, 2014, "Network Anomaly Detection: Methods, Systems and Tools," *IEEE Communications Surveys & Tutorials* , vol. 16, no. 1, pp. 303-336, First Quarter 2014.

[9] Erhan Guven Anna L. Buczak, 2016, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153 - 1176, First Quarter 2016.

[10] Chandrika Palagiri, Rasheda Smith, Boleslaw Szymanski, Mark Embrechts Alan Bivens, 2002, "Network-Based Intrusion Detection Using Neural Networks," in *Intelligent Engineering Systems through Artificial Neural Networks ANNIE-2002*, New York, pp. 1-6.

[11] D. Mutz, W. Robertson, F. Valeur C. Kruegel, 2003, "Bayesian event classification for intrusion detection," in *19th Annual Computer Security Applications Conference*, Las Vegas.

[12] E. Kirda, C. Kruegel, M. Balduzzi L. Bilge, 2011, "EXPOSURE: Finding malicious domains using passive DNS analysis," in *18th Annual Network and Distributed System Security Symposium*, San Diego.

[13] Mohammad Zulkernine, Anwar Haque Jiong Zhang, 2008, "Random-Forests-Based Network Intrusion Detection Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Application and Reviews)*, vol. 38, no. 5, pp. 649-659, September 2008.

[14] M. R. Patra M. Panda, 2007, "Network intrusion detection using Naive Bayes," *IJCSNS International Journal of Computer Science and Network Security*, vol. 7, no. 12, pp. 258-263, December 2007.

[15] Apache Spot. [Online]. http://spot.incubator.apache.org/

[16] David M. Blei, Andrew Y. Ng, and Michael I. Jordan, 2003 "Latent Diriclet Allocation," *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993-1022.

[17] Sqrrl. [Online]. https://sqrrl.com/

[18] Apache Metron: Real-time big data security. [Online]. http://metron.apache.org/

[19] G. Gardikis and et al., 2017, "SHIELD: A Novel NFV-based Cybersecurity Framework ," in *IEEE NetSoft*, Bologna.

[20] The SHIELD Project: Securing against intruders and other threats through a NFV-enabled environment. [Online]. https://torsec.github.io/shield-h2020/

[21] Tasnuva Mahjabin, Yang Xiao, Guang Sun, and Wangdong Jiang, 2017, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *SAGE International Journal of Distributed Sensor Networks*.

[22] Ahmed Almusawi and Haleh Amintoosi, 2018, "DNS Tunneling detection method based on multilabel support vector machine," *Hindawi Security and Communication Networks*.

[23] The SHIELD Consortium, 2018, "D3.2 Updated specifications, design and architecture for the vNSF ecosystem,".

[24] The SHIELD Consortium, 2018, "D4.2 Updated specifications, design and architecture for the usable information-driven engine,".

[25] ETSI OSM. Open Source MANO. [Online]. https://osm.etsi.org/

[26] ETSI, 2014, "ETSI GS NFV-MAN 001 v1.1.1 Network functions Virtualisation(NFV); Management and Orchestration,".

[27] The Linux Foundation. OpenDaylight Carbon. [Online]. https://www.opendaylight.org/what-we-do/current-release/carbon

[28] Ludovic Jacquin, Adrian L. Shaw, and Chris Dalton, 2015, "Towards trusted software-defined networks using a hardware-based Integrity Measurement Architecture," in *2015 1st IEEE Conference on Network Softwarization (NetSoft)*, London, UK. [Online]. http://ieeexplore.ieee.org/document/7116186/

[29] Cataldo Basile, Antonio Lioy, Christian Pitscheider, Fulvio Valenza, and Marco Vallini, 2015, "A novel approach for integrating security policy enforcement with dynamic network virtualization," in *1st IEEE Conference on Network Softwarization (NetSoft)*, London, UK.

[30] The Apache Software Foundation. Spark MLlib. [Online]. https://spark.apache.org/mllib/

[31] SlowLoris GitHub page. [Online]. https://github.com/gkbrk/slowloris

[32] BoNeSi GitHub page. [Online]. https://github.com/Markus-Go/bonesi

[33] Iodine GitHub page. [Online]. https://github.com/yarrick/iodine

[34] PhantomJS website. [Online]. http://phantomjs.org/

[35] Alexa Top 500. [Online]. https://www.alexa.com/topsites

[36] iPerf - The ultimate speed test pool for TCP, UDP and SCTP. [Online]. https://iperf.fr/

[37] Natacha Turck, Alexandre Hainard Xavier Robin, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller, 2011, "pROC: an open-source package for R and S+ to analyze and compare ROC curves," *BMC Bioinformatics*.

[38] John W. Tukey, 1977, *Exploratory Data Analysis.*: Addison-Wesley.