Toward A Holistic, Efficient, Stacking Ensemble Intrusion Detection System using a Real Cloud-based Dataset

Ahmed M. Mahfouz[1], Abdullah Abuhussein[2], Faisal S. Alsubaei[3], Sajjan G. Shiva[4]
Department of Computer Science, University of Memphis, Memphis, TN 38152, USA[1, 4]
Information Systems Department, St. Cloud State University, St. Cloud, MN 56301, USA[2]
Department of Cybersecurity, University of Jeddah, Jeddah 23890, Saudi Arabia[3]

***Abstract*—Network intrusion detection is a key step in securing today's constantly developing networks. Various experiments have been put forward to propose new methods for resisting harmful cyber behaviors. Though, as cyber-attacks turn out to be more complex, the present methodologies fail to adequately solve the problem. Thus, network intrusion detection is now a significant decision-making challenge that requires an effective and intelligent approach. Various machine learning algorithms such as decision trees, neural networks, K nearest neighbor, logistic regression, support vector machine, and Naive Bayes have been utilized to detect anomalies in network traffic. However, such algorithms require adequate datasets to train and evaluate anomaly-based network intrusion detection systems. This paper presents a testbed that could be a model for building real-world datasets, as well as a newly generated dataset, derived from real network traffic, for intrusion detection. To utilize this real dataset, the paper also presents an ensemble intrusion detection model using a meta-classification approach enabled by stacked generalization to address the issue of detection accuracy and false alarm rate in intrusion detection systems.**

***Keywords*—Intrusion detection system; IDS dataset; stacking ensemble ids; stacking; security; ensemble learning**

I. Introduction

With the exponential growth of network-based applications globally, there has been a transformation in the business models of organizations [1]. Cost reduction of both computational devices and the Internet have led people to become more technology dependent. As a result of the increasing use of computer networks, new risks have emerged [2]. Therefore, the process of enhancing the speed and precision of security mechanisms has become crucial. Although abundant new security tools have been developed, the rapid evolution of malicious actions continues to be a demanding matter, as their ever-evolving attacks continue to create huge threats to network security [3]. Classical security techniques—for instance, firewalls—are used as a first line of defense against security problems but remain unable to detect internal intrusions or adequately provide security countermeasures [4]. Thus, network administrators tend to rely predominantly on Intrusion Detection Systems (IDSs) to detect such network intrusive actions.

During the past decade, it has become clear that the trend of using the cloud services model in preference to the old on-premises model is increasing rapidly for many reasons [5]. For instance, the unique utilization/charging models offered by the cloud provider that gives customers the flexibility to adjust their expenses easily, based on their needs. Scaling processes would consume much more time, effort, and expense without the cloud model. With the cloud model, the Capital Expenditure (CapEx) is reduced to the minimum or removed. These elements are taken care of by the cloud provider, which reduces the time to market (TTM) of the services and facilitates hunting market opportunities. With these merits, and many more, adopting the cloud model enables the customer to focus on service development rather than infrastructure management, which helps in achieving customer satisfaction and maximizing revenue. However, using the cloud model comes with many implications and consequences, especially on the security side of the model. One such implication is the huge increase in the number of machines exposed to the Internet since the management of those remote servers, hosted over the cloud, by legitimate users, entails enabling remote access to the servers, which increases the number and kind of vulnerabilities that can be exploited by the attackers.

With the advances in the field of machine learning, studying the malicious traffic patterns and the attacker`s behavior for the purpose of developing detection and mitigation/reduction algorithms has become a hot area of research [6]. A vital building block of most of the machine learning techniques is the dataset that is used either in the training phase in case of unsupervised learning or the training and testing phases in case of supervised learning. Due to the significance of the dataset (as shown later), many studies have been devoted to generating such a dataset using different techniques and setups [7].

Most of the time, the datasets used in different studies depend on a simulated dataset due to the lack of publicly available real datasets of the network attacks [8]. This is mainly attributed to the fact that organizations are usually hesitant to publicly share technical information with others about their computing assets, such as applications, network layout, or other information that can be extracted/guessed from a dataset. Doing so risks exposing confidential and sensitive data about the organization's computing assets from security and business perspectives and costs a lot more than taking the risk of sharing. Another reason for the scarcity of real datasets is that they would reveal valuable information about the organization's Intrusion Detection System (IDS) if the machine learning algorithm is trained on the same dataset, and this could help intruders to

bypass it. Although resorting to a simulated dataset seems to be a good solution, it could result in less accurate algorithms when applied in real-world systems [9]. Aside from being simulated or real, attack datasets used for machine learning models have a conceptual problem, which is the imbalance issue since the attacker would be trying to hide his traffic in the normal user traffic. Another shortcoming in the existing datasets is that most of them are a bit outdated, and most of the efforts focus on the attacks, but not the pre- or post-attack (attacker`s behavior).

This paper produces a new network intrusion dataset based on real network attacks on up-to-date cloud-based infrastructure. It also offers an adaptive ensemble classifier model, which integrates the advantages of different Machine Learning (ML) classifiers for diverse kinds of attacks and achieves best results using ensemble learning. The proposed model uses a meta-classification method based on stacked generalization for network IDS. The advantage of ensemble learning is combining the predictions of numerous base estimators to expand generalizability and strength over that of a single estimator.

II. Related Work

A. Dataset

The effectiveness of any study, or the accuracy of any algorithm that uses a dataset, greatly depends on the dataset quality in terms of both being correctly labelled and being up to date and able to capture the latest attacks [10]. Also, the more data instances there are in the dataset, the greater the accuracy of the experiments and the generalizability of the model. Network attack datasets are constructed by system logs, network logs, network flows, and memory dumps. A novel technique called generative adversarial networks is used to train a generator to create the dataset [11]. The dataset could be built using real or simulated data. The work of one group of researchers [12] provides a comprehensive overview of the existing datasets by analyzing 715 research articles. They focus on three aspects: the origin of the dataset (e.g., real-world vs. synthetic), whether datasets were released by the researchers or not, and the types of datasets that exist. They conclude that 56.4% of the datasets are generated via experiments, while 36.7% are real data. Also, 54.4% of the studies use existing datasets, while the rest created their own, and only 3.8% of them released their datasets. In another research project [13], the authors provide a comprehensive overview of the most used available datasets. Based on their research, the main limitations of the current datasets can be summarized as follows:

- Some of the datasets are old, so they do not help with the recent types of attacks.
- The dataset is not labeled, making it useless for training supervised machine learning models, unless manually labeled, which can be cumbersome.
- The dataset is limited to specific types of attacks or targets specific applications, reducing its generality.
- The dataset is small and does not contain enough data to generalize the trained model.
- The dataset contains redundant data, which could lead to biased models.
- The dataset is completely generated in the lab, making it less representative of the real-world attacks.
- The dataset consists of an imbalanced amount of attack data and benign traffic.

To address the above limitations, one study [14] proposed a dataset approach called CIDD (Cloud Intrusion Detection Dataset) for masquerade attacks. They developed a log analyzer and correlator system to parse and analyze the data from the network. These parsed data are fed to the log analyzer and correlator for processing and marking. The analyzer correlates the user audits in network and host environments using user IP and audit time. Then it assigns user audits to a set of VMs (Virtual Machines) according to their login sessions time and the characteristic of the user task. Finally, it uses the attack and masquerade tables provided by the MIT group to mark the malicious records. The drawback of this dataset is that it lacks the representation of real network traffic as well as actual attack simulations. Moreover, it is outdated for the adequate evaluation of modern IDSs on current networks, regarding types of attacks and the network infrastructure.

In other research [15], the researchers developed a testbed to generate their dataset. The testbed is composed of different machines in a Windows domain and each machine has different types of agents to collect logs and send them to the logger. These machines also have scripts to enable the simulation of some types of attacks, pushed by the logger server, as well as the generation of the normal traffic. The logger server is equipped with the necessary applications to play different roles. Examples of these are an elastic search to collect logs from the whole system, a Mitre Caldera Server to simulate various types of attacks using the installed agents on the hosts, an IDS Suricata for identifying network attack signatures in traffic that is used for labeling the dataset, and others. Unfortunately, the proposed testbed also does not represent real-world network traffic and lacks the actual attacks representation.

B. Stacking Ensemble IDS

Ensemble learning based methods apply collections of ML procedures to obtain higher predictive performance than could be obtained from one classifier [16]. The core idea of ensemble methods is to combine several classifiers to exploit the power of each single algorithm used to obtain a more powerful classifier. Ensemble learning methods are mainly helpful if a problem can be split into subproblems so that each subproblem can be assigned to one module of the ensemble. Depending on the structure of the ensemble approach, each module can include one or more of the ML algorithms. During network attacks, because the signatures of different attacks are distinct from each other, having different sets of features as well as different ML algorithms to detect different types of attacks is preferable. A single IDS cannot address all types of input data or identify different types of attacks [17, 18]. Many researchers have shown that a classification problem can be solved with high accuracy when using ensemble models instead of single classifiers [19, 20, 21, 22].

III. Real Cloud-based Dataset

A. Dataset Collection Setup

To collect real attack traffic, a testbed was built on AWS (Amazon Web Services) and was run for 10 days between the 8th and the 18th of March 2021. The system consists of three main subsystems: the Sensors, which is used as a decoy to lure the adversaries to try the system, The Collector, which gathers the data from different sensors, and the Visualizer, which parses, analyzes, searches, and extracts the collected data.

1) The sensors subsystem: Sensors are servers that are intentionally exposed to the public network, pretending to offer something interesting for the attacker. A lot of effort has been made to create such technology leading to what is known as a honeypot [23]. Which is a data framework asset whose esteem lies in unauthorized or unlawful utilization of that asset, which means that honeypots derive their values from the threats using them [24]. Honeypots, as a security approach, differ from firewalls and intrusion detection systems in the sense that they are implemented somewhere in the network intentionally with the hope of attracting hackers. If they are built the right way, with the right precautions, then the more they are attacked and the smarter those attacks are, the more valuable the honeypots are. A honeynet is a collection of high interaction honeypots on a tightly controlled and highly monitored network. A honeypot can be one of the three types:

- Low-interaction honeypot - This kind of honeypot gives the intruders the illusion that the system is running some services so that it has no risks and requires fewer resources, but it is easily discovered by the attacker [25].
- Medium interaction honeypot - This kind is a little more interactive as it simulates some services and enables the attacker to run commands on the system [25].
- High interaction honeypot - This kind can be a separate network of real running services for the sole purpose of deflecting the attacker from the actual services, collecting his data, and studying his behavior. It requires more resources and can be risky, but the collected data can be more valuable [26].

Besides using the honeypots as decoys to capture the attacker's data, they can also be of great value to trick and deflect the adversaries from the actual system, giving the administrators of the attacked system more time to harden the system and apply the necessary patches. In real enterprise systems, honeynets can be deployed either before or after the organization`s firewall. When deployed before the firewall, they allow the most exposure to as many attacks as possible. On the other hand, they can be deployed behind the firewall for two reasons: first, to capture internal attacks originating from inside the organization by those who are trying to do things they should not be doing. This is important since internal traffic usually does not go through the firewall. Second, to give an early alert that the organization`s firewall or IDS might need to be tuned after it was successfully evaded by some non-legitimate attacker. In this experiment, the sensors subsystem is built as a honeynet of six honeypots to collect data from the attackers. Different honeypots have different purposes and run/simulate different services. In this section, a brief description of each honeypot is given.

- Dionaea: a low-interaction honeypot that captures attack payloads and malware. Dionaea listens on many different protocols, e.g., blackhole, epmap, ftp, http, memcache, mirror, mqtt, mssql, mysql, pptp, sip, smb, tftp, upnp.
- Cowrie: a medium/high-interaction honeypot that emulates SSH and Telnet services and gives the intruder the illusion of interacting with a real system and hence captures his actions against the system, e.g., commands and downloaded files. It works by running a fake filesystem with the ability to add/remove files where a full fake filesystem resembling a Debian 5.0 installation is included. It allows the addition of fake file contents so the attacker can “cat” files, such as /etc/passwd. Cowrie also gives the attacker the ability to download and upload files using wget/curl or sftp/scp and saves files downloaded for later inspection.
- Conpot: a low-interaction honeypot that is designed to work as a server-side industrial control system (ICS).
- AMUN: another low-interaction honeypot designed to capture malware that exploits server-based vulnerabilities. AMUN simulates a lot of protocols like RDP, SMP, telnet, FTP, and more to emulate many vulnerabilities e.g., Buffer Overflow, Buffer Overrun, and Stack Overflow.
- Snort: a honeypot, but it is used in this project as a sensor for the traffic. Snort is an intrusion prevention system that was developed by Cisco, who opened its source and made it available to the community.
- P0f: a tool that utilizes an array of sophisticated, purely passive traffic fingerprinting mechanisms to identify the players behind any incidental TCP/IP communications (often as little as a single normal SYN) without interfering in any way. P0f can recognize the operating system, measurement of system uptime, distance, and link type.

2) The collector subsystem: We used Modern Honey Network (MHN), a central server for the management and data gathering of honeypots [27]. MHN is the brain of the testbed as it facilitates the deployment of honeypots by wrapping all the necessary software for each honeypot in a script, collects data from sensors, and enables integration with the visualizer, as well as providing a RESTful API for integration with 3rd parties.

As shown in Fig. 1, MHN composed of two main components:

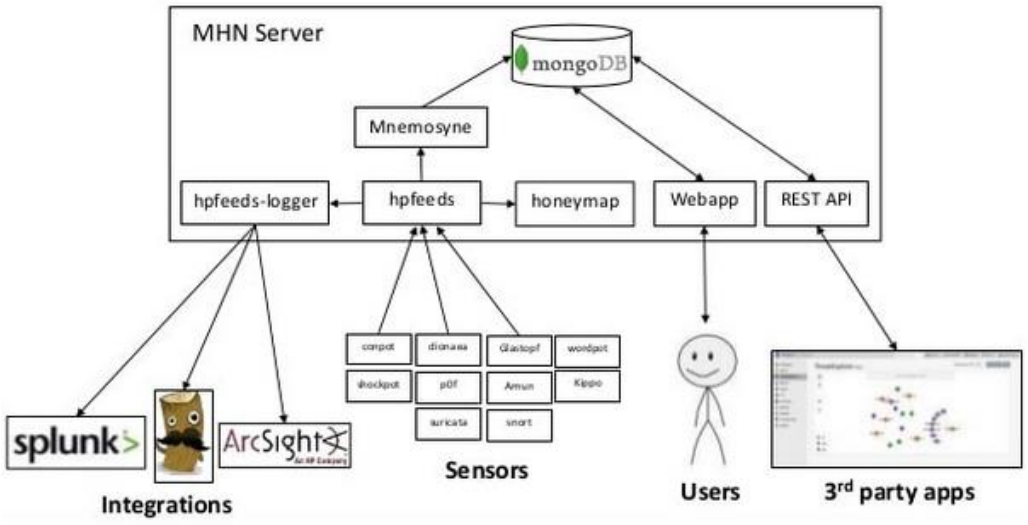


Fig. 1. MHN Server Architecture.

3) Lightweight authenticated publish-subscribe protocol (hpfeeds). It has a simple wire-format so that everyone is able to subscribe to the feeds with their favorite language in very little time, so it is used as the landing point from all the honeypots, and as a data source for three other system components:

a) Honey map, which is a fancy map to show the geographical location of live attacks from some types of honeypots like Dionaea.

b) Hpfeeds-logger is a simple utility for logging hpfeeds events to files compatible with Splunk and ArcSight.

c) Mnemosyne provides immutable persistence for hpfeeds. It also provides normalization of data to enable sensor agnostic analysis and exposes this normalized data through a RESTful API.

4) MongoDB is a general-purpose, document-based, distributed database used to store all the indexed data feed from Mnemosyne. The Mongo database is used as the data source for two other system components:

a) Web app, which is the basic built-in visualization component of MHN unless a more complex analysis is needed by a 3rd party like Splunk.

b) 3rd party API, which provides an API interface for 3rd party integration.

We built a testbed that consists of six sensors running different honeypots and one server running MHN and Splunk services. The honeypot servers were running on an AWS free tier t2-micro instance type, while the MHN & Splunk servers were running on a t2-medium instance during data collection, upgraded to t2-large instance type during data analysis and extraction.

5) The Visualizer: With the large amount of data collected by the sensors, it was better to use a third-party application to handle the data instead of the MHN built-in web app. Splunk is used in this project, but MHN also supports integration with ArcSight software.

Splunk is a software platform to search, analyze, and visualize the machine-generated data gathered from the websites, applications, sensors, and devices that make up the IT infrastructure and business. Splunk is a great tool when it comes to the processing of a huge amount of data, as it can provide real-time processing and accept any data input format, e.g., csv, and JSON. It can also be configured to give alerts about the machine's states and predict if resource scaling is needed. To make integration with other systems easy, Splunk has the concept of apps that are an extension/addon of Splunk functionality. This gives the developers of any applications, e.g., MHN, who want to use Splunk the ability to develop their own application with a customized user interface and visualization dashboards to serve a specific need. They may then upload it to the Splunk marketplace (splunkbase) to make it available for the Splunk community. This makes it easy for the users to integrate those applications with Splunk by just importing the application extension into Splunk, and occasionally doing a few setup steps like licensing and data source configuration. For MHN, there is an app with the same name that can be downloaded from the splunkbase.

B. The Dataset Collection Results

After the data was collected from the sensors by the MHN server and sent to Splunk for analysis and visualization, we used the Splunk query language, Splunk processing language (SPL), to extract the datasets. Table I summarizes the total amount of the collected data using the sensors subsystem, as well as the data collected per each sensor. In the section below, we present a sample of the dataset, a distribution of the data across the collection period, and a summary of the collected data per sensor.

By implementing a testbed hosted on Amazon's AWS cloud, we ran an experiment for 10 days and collected different attacks on different services. Using the data collected by different sensors, we created a real network attack dataset comprising

many interesting features that can be used to profile the attacker, e.g., source/destination IPs, source/destination port numbers (attacked service), ssh version, operating system name and version, link type, usernames and password tried by the attacker, tcp flags, ip ttl, and many more. A full list of the extracted features is shown in Table II. The dataset obtained can be used, or can be a seed, for a dataset that solves most of the common issues in the currently available datasets. It is real-world data by design, up-to-date, and can be kept up to date easily by running the testbed during specific periods. It can automate all the post-processing operations needed to get a ready dataset, thanks to the use of visualizers and query languages. The dataset represents different types of attacks and can easily represent more by deploying more honeypots.

Based on a 10-day experiment, the most attacked service was server message block (SBM), which might make sense as this service is used by the WannaCry attacks that have been spreading and active since 2017. SSH service comes second in the most attacked services as the attacker tries to exploit the lack of awareness of some users that use the default or weak credentials. The common username, "admin," was the most tried username and "password" came second as the most tried passwords, while the less expected, "nproc" (a bash command to get the total number of cores/threads on the machine) was the most tried password. The most used operating system by the attacker was Linux version 3 or later, while Windows came next, which makes sense as a lot of the hacking tools used are Linux-based, e.g., Kali. Although most of the attacks originated from the United States, it might or might not accurately reflect the actual attacker`s location since a serious attacker might be using compromised machines to mount his attacks. These could be located anywhere, or be using any cloud-hosted machines, which the US has most. The data showed that the top attacking single IP was in Panama and generated around 34,000 attacks during the 10 days. The most used command is "uname," which is used to get the operating system type, kernel version, and other information that is necessary to determine the suitable attacking scripts and tools. The second and the third most used commands are "echo" which is used to show whatever argument is used after it and "which ls" to get the full path of the "ls" command. The two commands might not be meant for actual use but just to check if this is a real system or a trap. This is good to know as it can guide the honeypot developers toward which commands, they need to simulate for a more deceptive honeypot.

TABLE I. TOTAL COLLECTED DATA

Sensor	Total Collected	Distinct SRC	Distinct SRC DEST_Port	Distinct SRC DEST Port SRC_Port
Dionaea	177,000	10,000	72,000	158,000
P0f	369,000	24,000	108,000	212,000
AMUN	245,000	9,000	10,000	228,000
Cowrie	58,000	1,243	1,243	45,000
Snort	108,000	6,200	53,000	67,000
Conpot	3,780	444	444	544
Total	960,780	50,887	244,687	755,544

TABLE II. THE FULL LIST OF EXTRACTED FEATURES

#	Feature Name	Description
1	_time	time of traffic capturing
2	app	honeypot captured the traffic
3	dest	dest ip
4	dest_port	dest port
5	dionaea_action	either Dionaea honeypot accept or reject the connection
6	direction	the direction of the captured traffic either in or out
7	eth_dst	the dest mac address
8	eth_src	the source mac address
9	host	Splunk server ip or hostname
10	ids_type	the type of the used ids
11	ip_id	the packet id
12	ip_len	packet length
13	ip_tos	packet type of service
14	ip_ttl	packet time to live
15	linecount	the number of lines of the captured traffic
16	p0f_app	protocol used by P0f for fingerprinting
17	p0f_link	the connection type at the attacker side like modem or dsl
18	p0f_os	the operating system of the machine generating the attack
19	p0f_uptime	how long since the attacking machine is up
20	protocol	tcp or udp
21	sensor	id assigned by MHN per honeypot
22	severity	severity rank of the attack
23	signature	the signature of the attack as matched by snort
24	snort_classification	a number given by snort to classify the traffic
25	snort_header	the rule header
26	snort_priority	assigns a severity level to rules
27	source	input data source (needed by Splunk)
28	sourcetype	input data type (needed by Splunk)
29	splunk_server	Splunk ip or hostname
30	src	attack src ip
31	src_port	attack source port
32	ssh_password	password used by the attacker trying to get ssh access
33	ssh_username	username used by the attacker trying to get ssh access
34	ssh_version	attacker ssh client version
35	tcp_flags	indicate a particular connection state or provide additional information
36	tcp_len	packet length
37	timeendpos	at which byte into the event the timestamp ends
38	timestartpos	at which byte the timestamp starts
39	transport	transport protocol type tcp or udp
40	type	honeypot event type
41	udp_len	packet length
42	vendor_product	name of the honeypot that captures the traffic
43	_raw	raw (not parsed) event

Analysis of the collected data showed some interesting findings for each sensor. Dionaea not only listens on the opened ports but also allows the attacker to download and upload files. Fig. 2 shows a list of the top downloaded binaries expressed as their MD5. It is worth noticing that the same files came from different sources. Splunk`s MHN application also adds a fast method to scan those files against different antiviruses via Total Hash and Virus Total websites. By clicking on the link, a web page will open automatically, search for the file, and show the scanning results, as shown below in Fig. 3. Fig. 4 to 8 show the most often attacked ports in different sensors. Fig. 9 shows the top link types, and Fig. 10 shows the operating systems. Fig. 11 shows the top URLs that were used by the attackers to download scripts, and binaries used to mount their attacks. Fig. 12 shows the top SSH versions, Fig. 13 shows the most used user/password pairs, and Fig. 14 shows the most often used attack commands. Fig. 15 shows the top attack types captured by Conpot.

IV. Stacking Ensemble IDS

This section presents an ensemble learning model using a meta-classification method enabled by stacked generalization. A newly generated dataset that was captured from real network traffic was used for experimentation. Observed results indicate that the proposed stacking ensemble can generate superior predictions having 95% accuracy.

A. Methodology

As illustrated in Fig. 16, the stacking model comprises base and metaclassifiers—namely, Neural Networks (NN), k nearest neighbor (KNN), Decision Tree (DT), and Support Vector Machine (SVM), respectively.

Authors in [28] illustrated that the integration of a set of single algorithms leads to optimum predictions. Stacking or stacked generalization is a concept proposed by Wolpert [29]. Several ML algorithms define their subjective biases on a learning set, ultimately filtering out biases. The implementation of a stacked model involves two kinds of sub-models, base (level 0 classifiers) and metamodels (level 1 or meta). The main logic of a stacking model lies in using the metaclassifier to predict the samples by studying the level 0 classifiers. Yan and Han [30] illustrated the great advantage of using the stacking models. They have stated that stacking can enhance prediction accuracy while working with unbalanced datasets. A study [31] was conducted to emphasize the application of AI-based classifiers. The researchers in that study explained that ensembles were able to adapt to the robust behaviors of malicious and normal traffic effectively. Algorithm 1 shows the entire classification process implemented in the classification framework involving multiple classifiers.

B. Data Pre-processing and Feature Selection

Pre-processing was utilized to handle different data found in the dataset. To eliminate noise, and fix inconsistencies found in the data, a statistical transformation tool is needed. In our proposed work, missing data and outliers were compensated for by making the distribution normal. However, lost values rely on singular features. While some features can be assigned zero as a missing value, others are assigned zero as an actual value where binary data are considered. To maintain such predicaments, consideration of relevant features that guarantee ideal expectations is essential. Thus, an integration of hashing and information gain (IG) was applied to extract the maximum desirable features. Feature scaling was also utilized to assure that those features possessing a greater numeric range did not dominate the ones in smaller numeric ranges. The dataset has many features but not all appear to be important. Consequently, only 11 features were chosen from the dataset. The fundamental features were assigned weights to prioritize them, and only the best features were extracted. The dimensionality of the features was reduced using a hashing approach. The chosen features are direction, eth_src, host, protocol, src, src_port, ssh_version, tcp_flags, tcp_len, type, and udp_len.

C. Classification

The methodology to actualize the classification system included the application of different classifiers to resolve the basic complexities of information found in both packet-based and flow-based datasets.

Fundamentally, KNN count on a distance function that calculates the similarity or variance between two network occurrences found in the datasets under consideration.

The Euclidean distance d(x, y) can be calculated by via the following equation:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(xi - yi)2} \quad (1)$$

where xi is the *ith* feature of the instance x, while *yi* is the *ith* feature of the instance y, and "n" is the whole number of features found in the dataset. Let C = C1, C2, C3, . . . C p. There are "p" labels in the dataset. Let "x" be the new sample to be predicted. The objective of KNN classifier is to determine "k" vectors that are close to x. If most of the vectors belong to class Cm, then x will be assigned the class label Cm.

The radial basis function (RBF) is a preferred kernel function for many classification problems in ML. The following equation defines the RBF:

$$(x, y) = \exp\left(-\frac{\|x - y'\|^2}{2\sigma^2}\right), \quad (2)$$

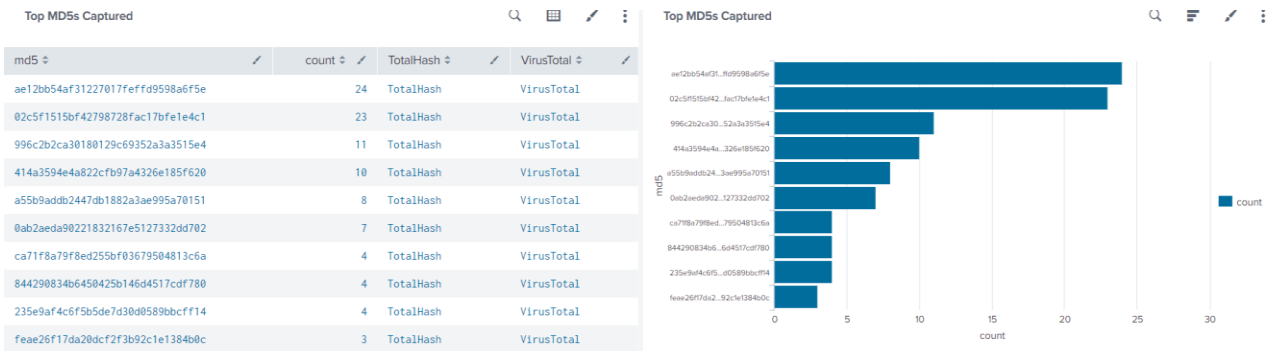


Fig. 2. Dionaea Top Captured MD5Binaries.

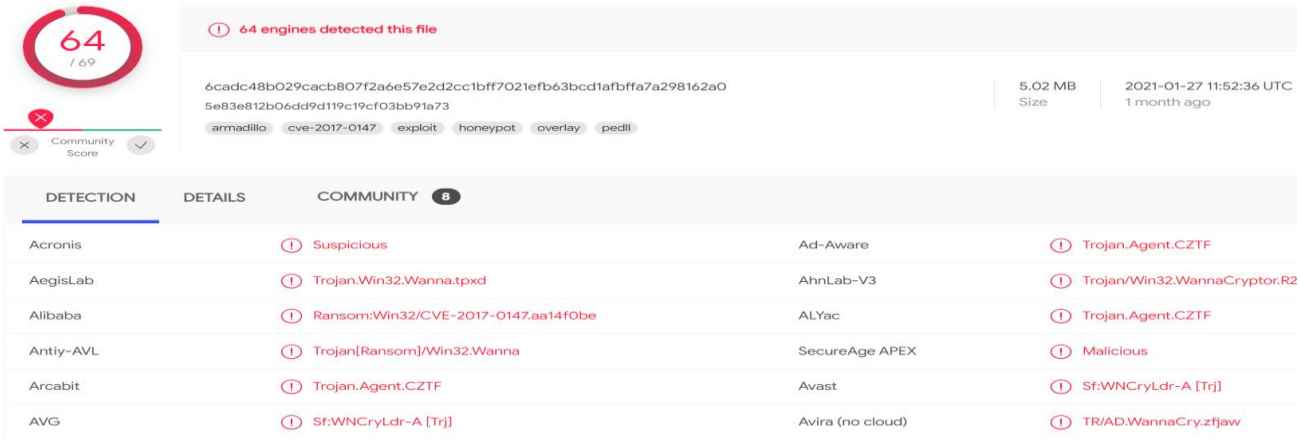


Fig. 3. Scanning Results for a Malware File.

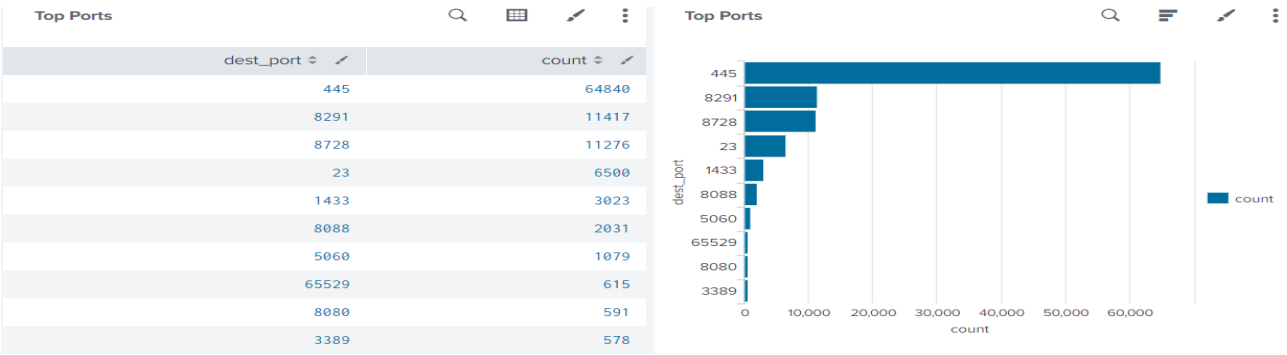


Fig. 4. Dionaea Top Attacked Ports.

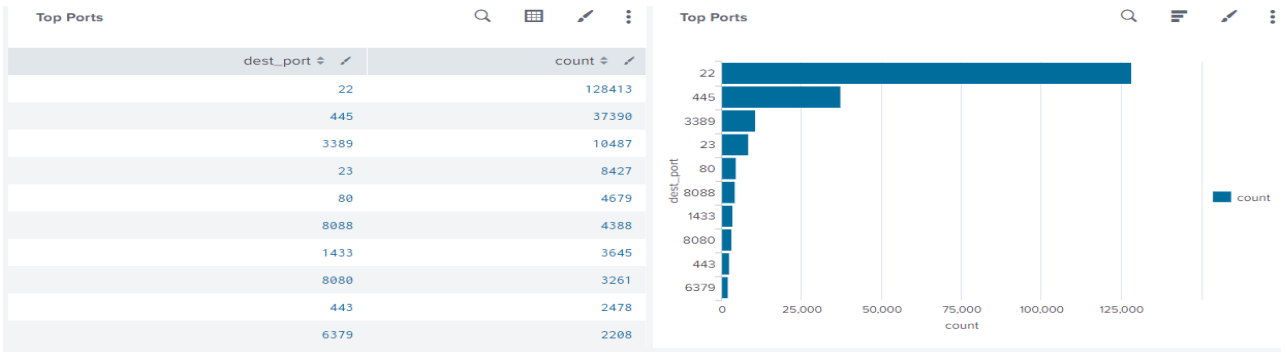


Fig. 5. P0f Top Attacked Ports.

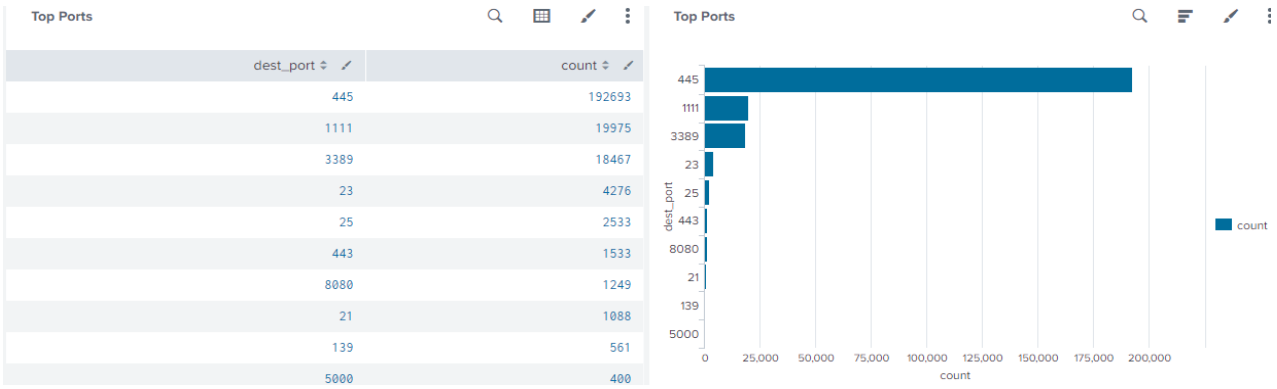


Fig. 6. AMUN Top Attacked Ports.

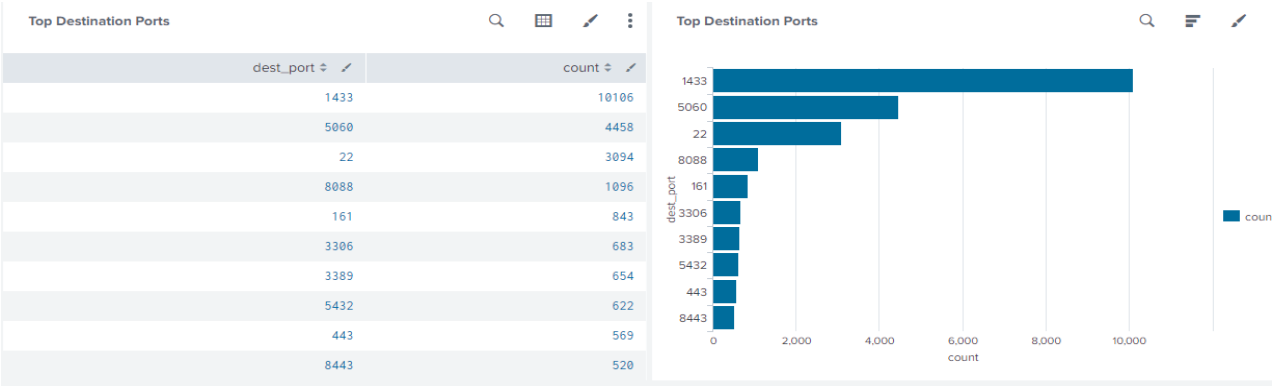


Fig. 7. Snort Top Attacked Ports.

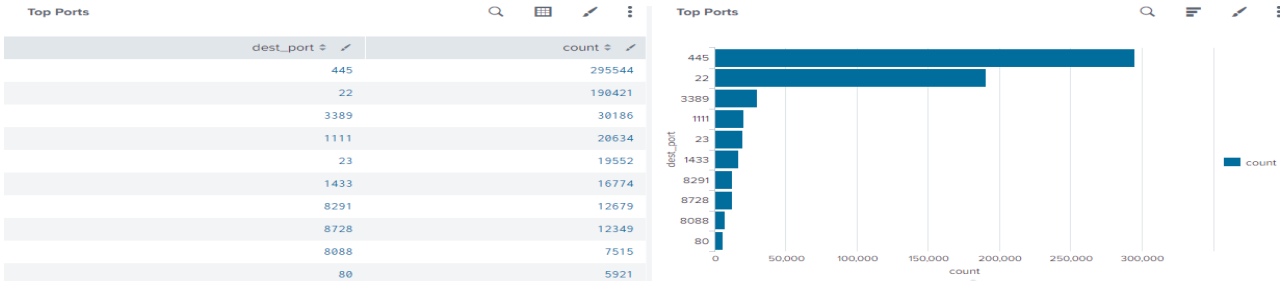


Fig. 8. Most Attacked Ports.

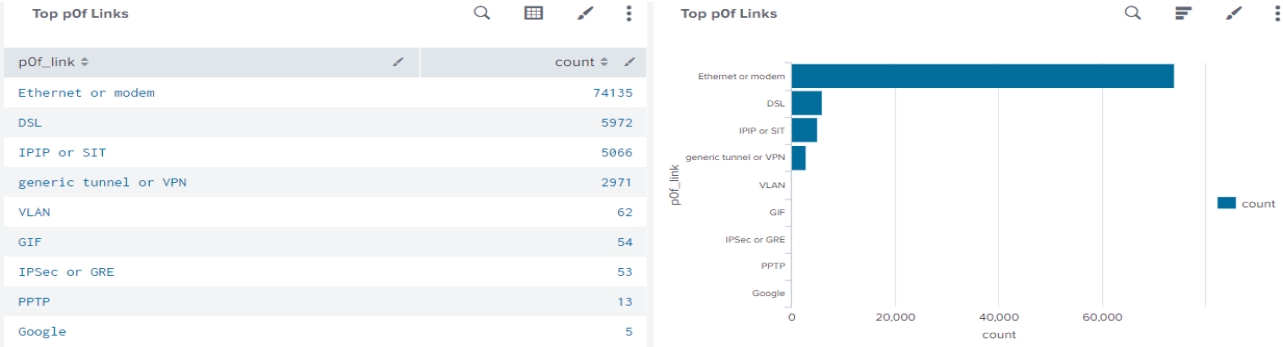
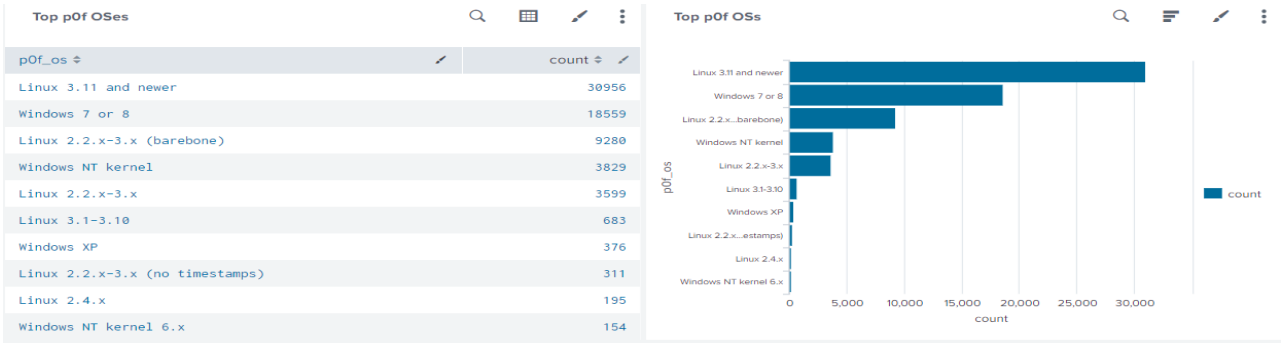


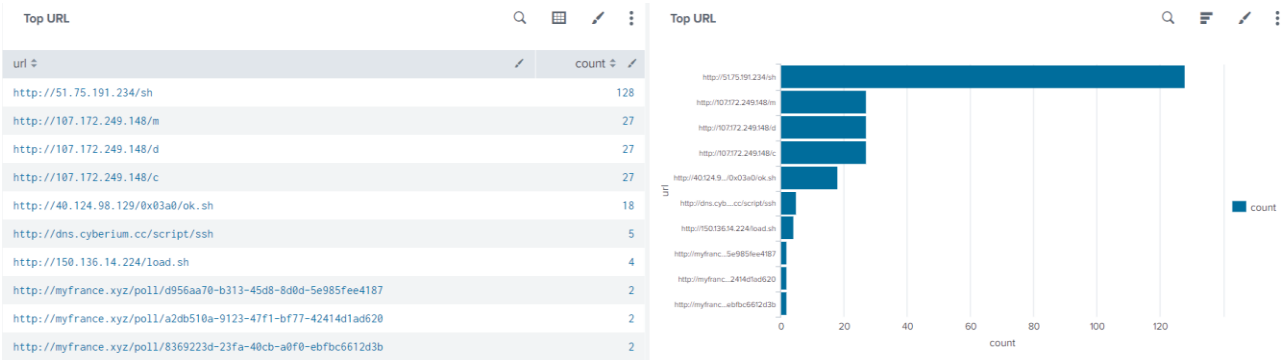
Fig. 9. P0f Top Link Types.



Top p0f OSes

p0f_os	count
Linux 3.11 and newer	30956
Windows 7 or 8	18559
Linux 2.2.x-3.x (barebone)	9280
Windows NT kernel	3829
Linux 2.2.x-3.x	3599
Linux 3.1-3.10	683
Windows XP	376
Linux 2.2.x-3.x (no timestamps)	311
Linux 2.4.x	195
Windows NT kernel 6.x	154

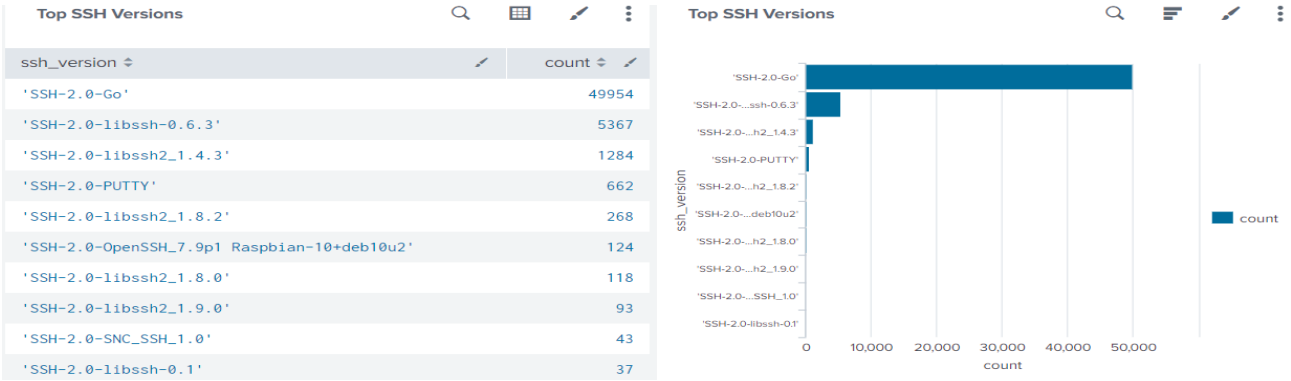
Fig. 10. P0f Top Operating Systems.



Top URL

url	count
http://51.75.191.234/sh	128
http://107.172.249.148/m	27
http://107.172.249.148/d	27
http://107.172.249.148/c	27
http://40.124.98.129/0x03a0/ok.sh	18
http://dns.cyberium.cc/script/ssh	5
http://150.136.14.224/load.sh	4
http://myfrance.xyz/poll/d956aa70-b313-45d8-8d0d-5e985fee4187	2
http://myfrance.xyz/poll/a2db510a-9123-47f1-bf77-42414d1ad620	2
http://myfrance.xyz/poll/8369223d-23fa-40cb-a0f0-ebfbc6612d3b	2

Fig. 11. Cowrie Top URLs.



Top SSH Versions

ssh_version	count
'SSH-2.0-Go'	49954
'SSH-2.0-libssh-0.6.3'	5367
'SSH-2.0-libssh2_1.4.3'	1284
'SSH-2.0-PUTTY'	662
'SSH-2.0-libssh2_1.8.2'	268
'SSH-2.0-OpenSSH_7.9p1 Raspbian-10+deb10u2'	124
'SSH-2.0-libssh2_1.8.0'	118
'SSH-2.0-libssh2_1.9.0'	93
'SSH-2.0-SNC_SSH_1.0'	43
'SSH-2.0-libssh-0.1'	37

Fig. 12. Cowrie Top SSH Versions.

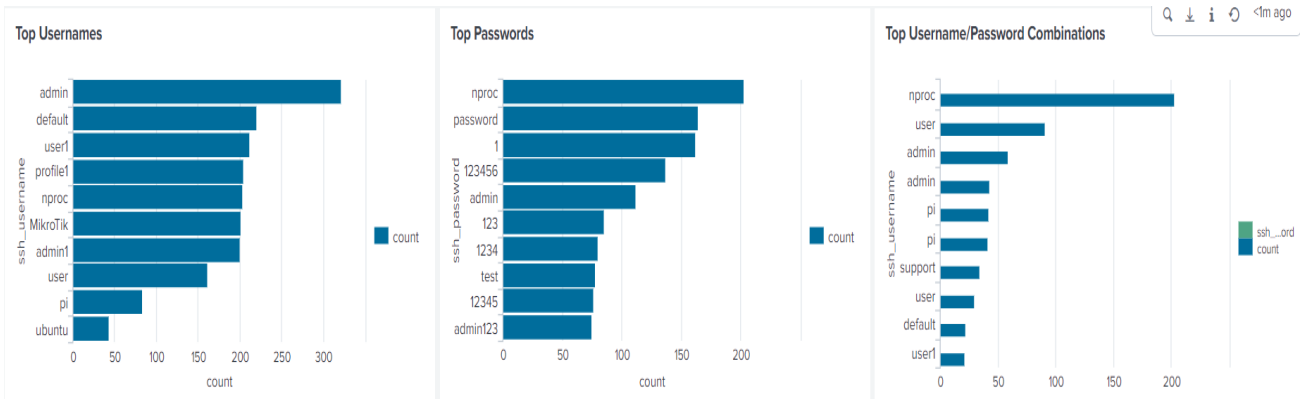


Fig. 13. Cowrie Top Users/Passwords.

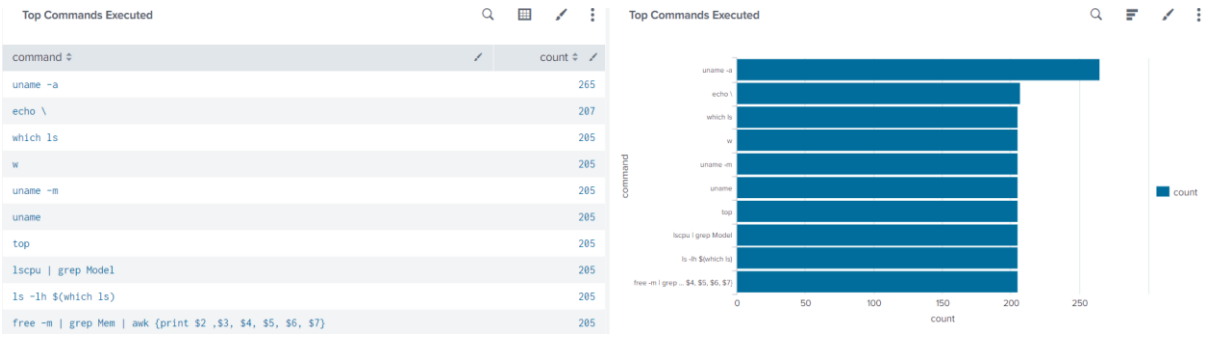


Fig. 14. Cowrie Top Attack Commands.

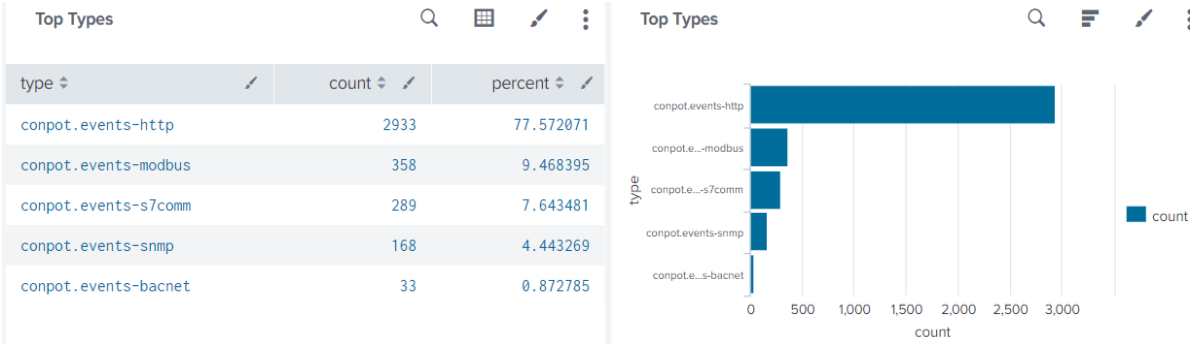


Fig. 15. Conpot Captured Top Attack Types.

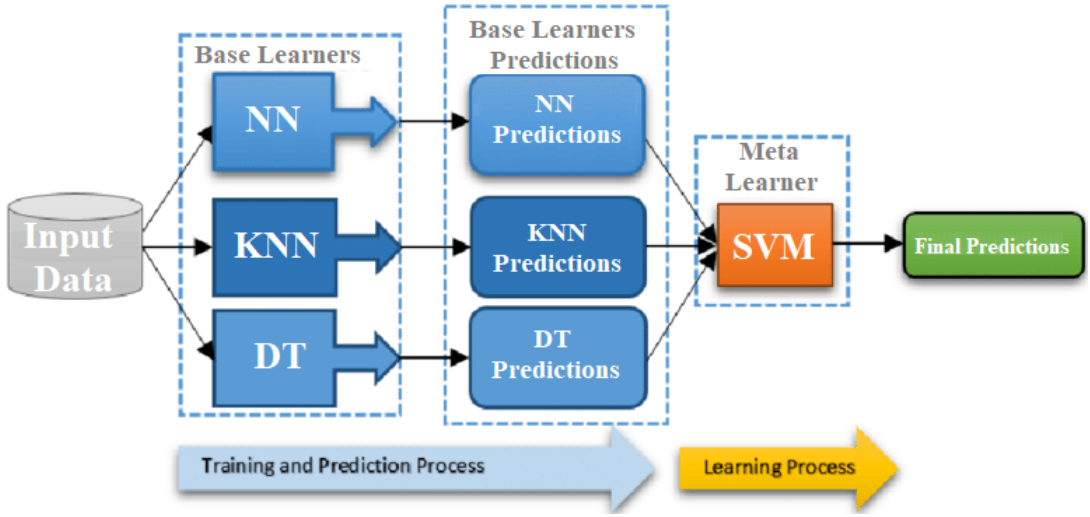


Fig. 16. Stacking Ensemble Model.

Algorithm 1: Stacking Ensemble Strategy.

Input: Train data $T = \{X_i, Y_i\}_{i=1}^{m}$ $(X_i \in R^n, Y_i \in Y)$
Output: Predictions from the ensemble E
Step 1. Impose cross validation in order to prepare a training set for meta-classifier
Step 2. Randomly split T into "m" equal size subsets, i.e., $T = \{T_1, T_2, T_3 \ldots T_m\}$
Step 3. for $m \longleftarrow 1$ to M
  Learn base classifiers namely NN, KNN, and DT
  for $n \longleftarrow 1$ to N
    Learn a classifier P_{mn} from T or T_m
  End for
Step 4. Formulate a training set for metaclassifier (SVM)
  for each $X_i \in Tm$
  Extract a new instance (x_i', y_i), where $x_i' = \{P_{m1}(X_i), P_{m2}(X_i), P_{m3}(X_i), \ldots, P_{mN}(X_i)\}$
  End for
End for
Step 5. Return $y_q = \{y_1, y_2, y_3, \ldots, y_n\}$ from ensemble

Algorithm 1 shows the whole classification process implemented in the classification framework involving multiple classifiers.

D. Results and Discussion

The quality of any IDS can be measured by four performance metrics: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

To accurately value the performance of the proposed approach and assure the results acquired from the stacked ensemble model, both binary and multiclass classification results are given in this section. Table III depicts the results acquired upon classifying the network instances of the dataset into either attack or normal. Moreover, to test the predictions and to assure that the models do not overfit, mean training accuracy (MTA), mean training precision (MTP), and mean training recall (MTR) values are also mentioned in Table IV.

The NetFlow traces found in the dataset contain genuine background network traffic for a substantial duration of ten days. As per the confusion matrix presented in Table V, all seven attack types found in the dataset were distinguished perfectly aptly by the stacking classifier.

The proposed ensemble model could identify the occurrence of SSH scan attack in the foremost effective way. In order to show reliable results, performance metrics like precision and recall were also considered in addition to accuracy. Recall is the ability of the intrusion detection model to correctly locate the positive instances, where precision is the model's capability to locate the percentage of positive instances that were identified correctly.

Table VI shows that the false alarm rate is extremely least regarding all attack classes, which is a signal that the general effectiveness of the proposed ensemble model is very good. The ROC curve also is shown in Fig. 17.

TABLE III. BINARY CLASSIFICATION RESULTS

Accuracy	Precision	Recall	F1 score	AUC	FAR (%)
0.94	0.96	0.93	0.95	0.99	5.2

TABLE IV. TRAINING RESULTS OBTAINED BY 10-FOLD CROSS VALIDATION

Fold Number	Training Accuracy	Training Recall	Training Precision
1	0.9289	0.9142	0.9488
2	0.9299	0.9129	0.9520
3	0.9239	0.9101	0.9469
4	0.9278	0.9102	0.9500
5	0.9301	0.9109	0.9531
6	0.9319	0.9129	0.9480
7	0.9430	0.9040	0.9481
8	0.9258	0.9089	0.9519
9	0.9290	0.9140	0.9479
10	0.9260	0.9110	0.9509
	MTA: 0.9285	MTR:0.9115	MTP: 0.9497

TABLE V. CONFUSION MATRIX OF ALL THE 7 ATTACK TYPES

		0	1
SSHscan	0	0943204	05809
	1	07824	091829
UDPscan	0	0891295	016466
	1	018477	0121338
Spam	0	0932187	09632
	1	013268	093589
DOS	0	0936949	013311
	1	09348	089968
Scan	0	0927854	011710
	1	010253	099759
Blacklist	0	0940476	09738
	1	08962	089420
DDOS	0	0927785	014583
	1	060303	046915

TABLE VI. CLASS-WISE PERFORMANCE

Metric	Blacklist	Spam	Scan	SSH scan	UDPscan	DOS	DDOS	Overall
Recall	0.9918	0.98597	0.98907	0.99056	0.9797128	0.99012	0.93897	0.9809
Precision	0.9940	0.98988	0.98859	0.98976	0.9818808	0.98703	0.98557	0.9881
FAR	0.0054	0.0062	0.0102	0.0093	0.0147	0.0117	0.0130	0.0101
Accuracy	0.9871	0.97826	0.98001	0.98218	0.9666758	0.97934	0.92954	97.19%

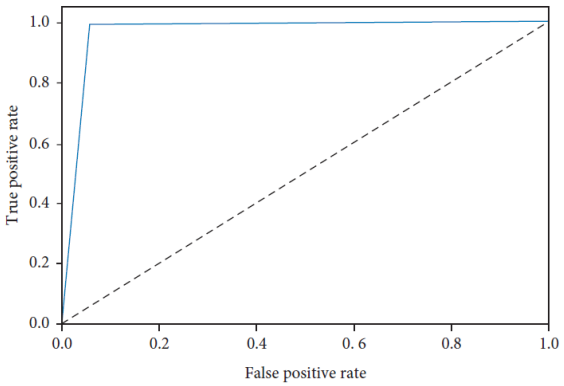


Fig. 17. ROC Curve.

V. CONCLUSION AND FUTURE WORK

This paper has presented an ensemble methodology based on a newly generated dataset that was extracted from real network traffic. The extensive dataset that has been created provides valuable benefits for training ML models to detect current attack types efficiently and accurately. This is because it overcomes most deficiencies of the present available datasets and covers most of the essential standards with common updated attacks. Moreover, the produced dataset is fully labeled and includes different network traffic features that are extracted and calculated for all normal and intrusion flows.

To utilize the created dataset, we presented an adaptive stacking ensemble learning model that integrates the advantages of different ML algorithms for diverse kinds of attacks and achieves optimal results through ensemble learning. This combines the predictions of several base estimators (i.e., NN, KNN, DT, and SVM) to accelerate the processing speed and improve scalability with a larger amount of network traffic data. The experimental results have shown that the ensemble model was able to enhance the classification accuracy, increase the true positive rate, and decrease the false positive rate. The real dataset provided can help cybersecurity researchers and firms to better understand the recent networking environment traffic, and traits of recent attacks, in order to better detect and prevent them. It can also help law enforcement and digital forensics teams in investigating cyberattacks. The proposed ensemble model can also be utilized with the provided dataset as a training dataset to detect and classify potential network attacks. This can help service providers, like cloud service providers, to monitor and improve their infrastructure.

This work can be expanded in the future to cover more and/or new attacks by collecting more networking traffic in different environments such as the Internet of Things networks, fog, etc. In addition, we can investigate the effectiveness of the ensemble model against such new networking traffic and suggest different features and tuning for every type of environment. More experimental analysis and a complete comparison with literature would be considered as well.

VI. DATA AVAILABILITY

The dataset generated in this work is publicly available and can be accessed from this link. https://www.researchgate.net/publication/356809493_Towards_A_Holistic_Efficient_Stacking_Ensemble_Intrusion_Detection_System_Using_Real_Cloud-based_Dataset.

REFERENCES

[1] Libert, B., M. Beck, and J. Wind, The network imperative: How to survive and grow in the age of digital business models. 2016: Harvard Business Review Press.

[2] Neumann, P.G., Computer-related risks. 1994: Addison-Wesley Professional.

[3] Demestichas, K., N. Peppes, and T.J.S. Alexakis, Survey on security threats in agricultural IoT and smart farming. 2020. 20(22): p. 6458.

[4] Cheminod, M., L. Durante, and A.J.I.t.o.i.i. Valenzano, Review of security issues in industrial networks. 2012. 9(1): p. 277-293.

[5] Gorelik, E., Cloud computing models. 2013, Massachusetts Institute of Technology.

[6] Haq, N.F., et al., Application of machine learning approaches in intrusion detection system: a survey. 2015. 4(3): p. 9-18.

[7] Moustafa, N. and J. Slay. The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems. in 2015 4th international workshop on building analysis datasets and gathering experience returns for security (BADGERS). 2015. IEEE.

[8] Shiravi, A., et al., Toward developing a systematic approach to generate benchmark datasets for intrusion detection. 2012. 31(3): p. 357-374.

[9] Li, Y.-F., et al., A systematic comparison of metamodeling techniques for simulation optimization in decision support systems. 2010. 10(4): p. 1257-1273.

[10] Khraisat, A., et al., Survey of intrusion detection systems: techniques, datasets and challenges. 2019. 2(1): p. 1-22.

[11] Xie, H., K. Lv, and C. Hu. An effective method to generate simulated attack data based on generative adversarial nets. in 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). 2018. IEEE.

[12] Grajeda, C., F. Breitinger, and I.J.D.I. Baggili, Availability of datasets for digital forensics–and what is missing. 2017. 22: p. S94-S105.

[13] Devi, M.G. and M.J. Nene. Scarce Attack Datasets and Experimental Dataset Generation. in 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA). 2018. IEEE.

[14] Kholidy, H.A. and F. Baiardi. Cidd: A cloud intrusion detection dataset for cloud computing and masquerade attacks. in 2012 Ninth International Conference on Information Technology-New Generations. 2012. IEEE.

[15] Nazarov, A., A. Sychev, and I. Voronkov. The Role of Datasets when Building Next Generation Intrusion Detection Systems. in 2019 Wave Electronics and its Application in Information and Telecommunication Systems (WECONF). 2019. IEEE.

[16] Dasgupta, D., et al., Machine learning in cybersecurity: a comprehensive survey. 2020: p. 1548512920951275.

[17] Shalev-Shwartz, S. and S. Ben-David, Understanding machine learning: From theory to algorithms. 2014: Cambridge university press.

[18] Jordan, M.I. and T.M.J.S. Mitchell, Machine learning: Trends, perspectives, and prospects. 2015. 349(6245): p. 255-260.

[19] Fayyad, U.M. and K.B.J.M.l. Irani, On the handling of continuous-valued attributes in decision tree generation. 1992. 8(1): p. 87-102.

[20] Diplaris, S., et al. Protein classification with multiple algorithms. in Panhellenic Conference on Informatics. 2005. Springer.

[21] Oza, N.C. and K.J.I.f. Tumer, Classifier ensembles: Select real-world applications. 2008. 9(1): p. 4-20.

[22] Chand, N., et al. A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection. in 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring). 2016. IEEE.

[23] Perkins, R.C. and C.J. Howell, Honeypots for Cybercrime Research, in Researching Cybercrimes. 2021, Springer. p. 233-261.

[24] Spitzner, L. Honeypots: Catching the insider threat. in 19th Annual Computer Security Applications Conference, 2003. Proceedings. 2003. IEEE.

[25] Almotairi, S., et al. A technique for detecting new attacks in low-interaction honeypot traffic. in 2009 Fourth International Conference on Internet Monitoring and Protection. 2009. IEEE.

[26] Nicomette, V., et al., Set-up and deployment of a high-interaction honeypot: experiment and lessons learned. 2011. 7(2): p. 143-157.

[27] Wafi, H., et al. Implementation of a modern security systems honeypot honey network on wireless networks. in 2017 International Young Engineers Forum (YEF-ECE). 2017. IEEE.

[28] Van der Laan, M.J., et al., Super learner. 2007. 6(1).

[29] Wolpert, D.H.J.N.n., Stacked generalization. 1992. 5(2): p. 241-259.

[30] Yan, J. and S.J.M.P.i.E. Han, Classifying imbalanced data sets by a novel re-sample and cost-sensitive stacked generalization method. 2018. 2018.

[31] Kumar, G., K.J.A.C.I. Kumar, and S. Computing, The use of artificial-intelligence-based ensembles for intrusion detection: a review. 2012. 2012.