Assessing the Effectiveness of Snort in Detecting Malicious URLs

by

Simbarashe Zuva

B.Tech, Parul University, Vadodara, 2020

A Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

University of Victoria

Assessing the Effectiveness of Snort in Detecting Malicious URLs

by

Simbarashe Zuva

B.Tech, Parul University, Vadodara, 2020

Supervisory Committee

Dr. Issa Traore, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Isaac Woungang, Co-Supervisor
(Department of Electrical and Computer Engineering)

Abstract

Web attacks have been on the rise in recent years, and organisations are constantly searching for new and better ways to detect and block the corresponding attack vectors. Some of the prominent attributes of web attack vectors are malicious domains used to trigger or sustain these attacks, for instance, through launching phishing attacks or by hosting command and control (C&C) infrastructures. Detecting accurately and blocking the malicious domains has become increasingly difficult due to the evasive techniques used by the attackers to mask their activities by emulating legitimate network traffic to an accurately high degree and through tactics such as domain generation algorithms (DGA) and fast flux DNS. Snort, an open-source intrusion detection system, has traditionally been utilized to detect network intrusions through network traffic signature analysis. However, while snort has subsequently been upgraded to enable the detection of web attacks, its effectiveness in detecting malicious domains is questionable because of the coarse-grained nature of web attack signatures. At the same time, it is a reasonable proposition to assume that there would be an implicit relation between granular attacks and the usage/occurrence of malicious domains. In this project, a platform is developed to explore and assess experimentally the ability of snort in detecting malicious domains. The proposed approach extracts some useful indicators of compromise (IoC) from the granular Snort alerts triggered by web visits and leverage such information to establish whether the corresponding URLs are benign or malicious. The platform was built around a headless chrome browser and the pfSense open-source firewall which has a built-in snort engine. The experimental evaluation, conducted using a public dataset of benign and malicious

domains, yielded important insights into the strengths and limitations of snort in detecting malicious domains, and helped identify directions for future improvements.

Table of Contents

List of Figures

List of Tables

Glossary

NIDS: Network Intrusion Detection Systems

IDS: Intrusion Detection System

IPS: Intrusion Prevention System

GUI: Graphical User Interface

IoC: Indicators of compromise (IoC)

HTTPS: Hyper Text Transfer Protocol - Secure

MitM: Man-in-the-Middle

CSRF: Cross-site request forgery

XSS: Cross-site scripting

DNS: Domain Name System

DDoS: Distributed Denial of Service

URL: Uniform Resource Locator

TPR: True Positive Rate

TP: True Positive

FN: False Negative

FP: False Positive

TN: True Negative

SQL: Structured Query Language

OS: Operating System

CSV: Comma Separated Variable

Acknowledgments

I would like to express my heartfelt gratitude to Dr. Issa Traoré and Dr. Isaac Wougang, my supervisors, for their unwavering support, patience and generous assistance throughout the completion of all the work surmounting to this project.

I also want to thank family my friends, colleagues and classmates for their support throughout this journey.

Dedication

I would like to respectfully dedicate this report to the memory of my beloved grandmother.

Chapter 1: Introduction

1.1 Context

The Internet has been a catalyst for the growth of businesses, healthcare, and many other applications, and it has transformed the way that people interact. However, with the rise of its use and the integration in different real-life faculties, there has been an increase in the number of cyberattacks. These attacks have become common, and their complexity has grown over the recent years, making them more challenging to be detected and/or mitigated. The use of intrusion detection systems (IDSs) has been advocated as an effective approach to detect and mitigate cyber-attacks.

1.2 Objectives

Snort is a popular IDS which is known primarily for its ability to detect intrusions by analysing the network traffic using a set of predefined and custom rules which encode known attack patterns or signatures. While Snort has an established track record for network intrusion detection, it is only recently that web attack detection rules have started to be incorporated in its rule engine. As a matter of fact, the effectiveness of Snort in detecting web attacks has not been proven or established. In this project, the effectiveness of Snort in detecting domains that carry out phishing-related attacks and malware, is evaluated. Additionally, some improvements that can be made to the system to enhance its capabilities, are presented. The evaluation is performed using the web-based GUI provided by pfSense, in detecting web attacks, in this case phishing related activities.

This thesis is a continuation of the work initiated in [3], but here, the effectiveness of Snort in detecting malicious URLs is assessed and validated through experiments.

1.3 Approach

Because of the prominent role of malicious domains in the conduct of cyberattacks, our objective in this project was to explore whether Snort alerts can be associated with or linked to visits to such domains. Traditional Snort signatures are coarse-grained in the sense that they are designed to cover some common and broad scope cyberattacks such as cross-scripting, SQL injection, denial of service, to name a few. It is not practical to design the signatures focusing particularly on detecting the malicious domains because of the large number and wide diversity of these domains. Doing so would require a finer level of granularity in designing the signatures, which is intractable. However, it is reasonable to expect that there would be an implicit relation (or link) between the broad scope of attacks detected by Snort and the occurrence of visits to the malicious domains. In this project, I focused on testing empirically such proposition. The naïve approach to conduct such test would have been to submit a URL (malicious or benign) to a web browser and observe the response generated by Snort in terms of alerts or lack of them. However, for this response to be statistically significant, it is needed to repeat such test over many URLs, which requires some form of automation. To automate the process on a large scale, I used a headless Chrome browser, which allows submitting the URLs either through a command line or programmatically using a script. I implemented a script that reads the URLs from a

CSV file, submit them in batch to the headless browser, which attempts to load the corresponding pages.

By using the Selenium web driver, the URLs in the dataset can be opened in batches i.e., multiple domains are requested simultaneously through the Chrome browser in headless mode. This is helpful for Snort to fully examine the network traffic with all its content data without the user manually opening each of the domains in question one by one. Snort monitors the traffic for anomalies while Wireshark monitors and displays the network traffic for which the data will be used for evaluating the Snort alerts. The number of domains in each batch are at the user's discretion but is heavily dependent on the computational resources that are available.

In the backend, Snort analyzes the generated network traffic and triggers some alerts if applicable. By analyzing the alerts, I can determine whether these alerts were false or true positives based on the labels of the URLs.

A key challenge faced in the assessment is the fact that many domains, particularly in the malicious category, did not resolve to valid IPs. This was because the datasets were relatively old, and the malicious domains tend to be short lived. To address this challenge, the data from Wireshark was preserved and repurposed for matching the domains with the IP addresses which were working at the time of the experiment. This ensures the correspondence with the results obtained using Snort, and therefore the risks of errors are reduced. The extraction of the IPs was done by the merging and string-matching steps of our approach using some criteria and methods (as explained in Chapter 3, Section 3.2).

1.4 Report Outline

This report is structured is as follows:

- Chapter 2 provides a background on Snort, including relevant basic knowledge of Kali Linux, Deepin and pfSense, malicious domain classification, characteristics, and potential risks.
- Chapter 3 presents the proposed approach and discusses the architecture and components of the platform deployed to assess the effectiveness of Snort in detecting malicious domains.
- Chapter 4 describes the datasets used in the project, and presents the experimental procedure, performance metrics, and performance evaluation results.
- Chapter 5 makes some concluding remarks and suggestions for future work.

Chapter 2: Background

2.1 Web Attacks

Web attacks can be broadly classified into three categories: client-side attacks, server-side attacks, and hybrid attacks. Client-side attacks exploit vulnerabilities in the client's browser or user interaction while server-side attacks exploit vulnerabilities on the server-side code. Hybrid attacks are a combination of both client-side and server-side attacks [51]. Common examples of web attacks include cross-site scripting (XSS), SQL injection, path traversal and command injection [10][11].

2.2. Platforms and Tools

Intrusion Detection Systems (IDSs) are systems that monitor the network traffic to detect the signs of malicious activity. Snort is an open-source IDS that is widely used due to its effectiveness and flexibility. It is designed to detect and prevent attacks in real-time using the attack signatures under some form of rules. Traditionally, it works through a Command Line Interface (CLI) and the rules must be manually imported, and customized rules have to be written down as local rules in a file located in the local subfolder of Snort.

Wireshark is a popular tool used to capture and analyze the network traffic. Such traffic is saved in the form of a packet file for simulations purpose and can be exported as a text-based dataset in the form of a CSV file. The versatility of Wireshark allows the user to choose which attributes to obtain from the network traffic. Examples of attributes are packet's address (both resolved and unresolved), header length, message status, to name a few.

Kali Linux is a Debian-based Linux distribution that is well renowned in the pen-testing community given the security testing tools it comes with by default. Kali started off as BackTrack developed by Offensive security based on the Knoppix Linux OS [52].

Selenium web driver is a tool used for web testing that works by emulating the standard browser functionalities without the GUI component. This allows web testers to examine the attributes of a webpage without going through cosmetic features, therefore allowing a faster browsing without trading off other web features such as encoded scripts.

Deepin is a Debian-based Linux distribution that has almost the same look and feel as Kali Linux with less security tools. Deepin is more GUI-oriented and user friendly compared to Kali and other popular Kali distros that are used for pen testing.

PfSense is an open-source firewall and routing software developed by Rubicon Communications, LLC. It comes bundled with the Unix-based FreeBSD operating system. Its web GUI is based on the Apache license, and it uses PHP webpages for accessing the configurations. It is also a comprehensive network security solution with the capabilities of integrating multiple tools into one centralized security solution [13]. In the same line, Snort is included as a downloadable package and its configurations can be modified through either the actual snort configuration file (snort.conf) or through the web GUI. The web GUI also allows the user to add some customized rules, tweak the policy modes and select some specific rules on a network. Due to its scalability and flexibility, pfSense seems preferrable to most off-the-cloud security implementations of similar nature [14], [15]. The reason for choosing pfSense instead of Kali Linux as a platform for Snort was because of its GUI capabilities which help in rule management as the experiment would be focused on domain detection rather than network intrusion. This calls for pruning out rules and keep

the ones applicable to this experiment that is easy to follow and trace back to in cases of failure or improvement by way of backing up configurations. This contrasts with the conventional Snort setup that requires the user to manually comment out and move irrelevant rules which is highly prone to error and difficult to follow up on for editions. pfSense has four different IPS policies, which are a preset collection of rules that are used with varying degrees of security strictness. The WAN categories page under the Snort interface configurations in pfSense's web configurator defines four different IPS policies termed as *Connectivity, Balanced*, *Security*, and *Max-Detect*.

Connectivity policy blocks most major threats with almost no false positive. Balanced encompass all rules in Connectivity, and in addition, it is speedy and it covers most recent threats. Security is more stringent and additionally it encompasses all the rules in the first two policies. Max-Detect is meant to be used for testing the network traffic in development systems, not in production environment.

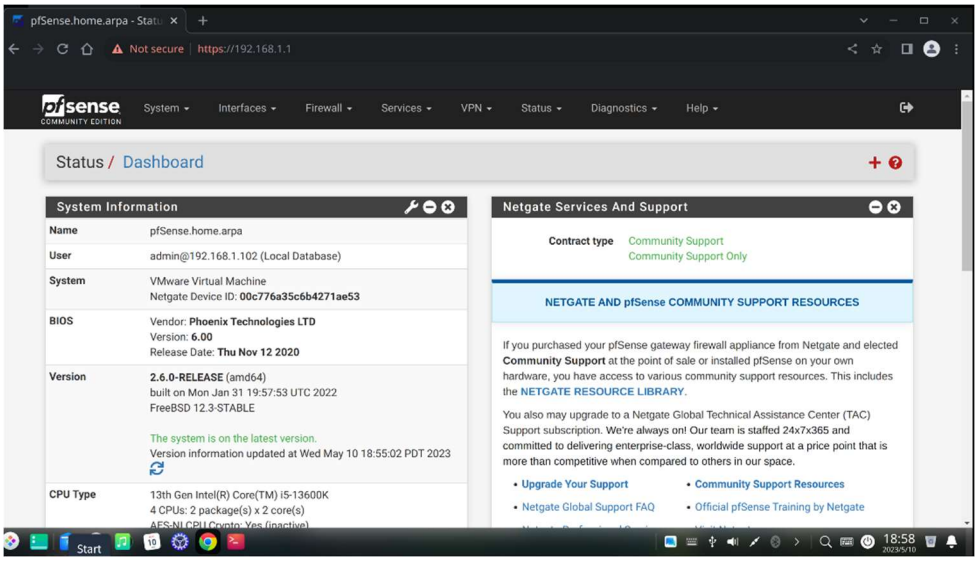

Fig 2.1 pfSense Overview

Figure 2.1 shows the dashboard of a newly installed pfSense firewall in functional state (whose easy indication of functionality is the green text in the version information on the system information) through the web configurator using Google Chrome in Deepin. It is important to ensure the connectivity within the pfSense and across the local network by disabling the gateway monitoring action. Adding Snort is done through the package manager available under the System tab.

2.3 Classification of Malicious Domains

In the context of this project, the classification of the domains is limited to two classes: Phishing domains and Legitimate domains.

Phishing domains: these websites are created and exist to steal sensitive information from the unsuspecting victims. This may lead to a fraudulent activity, thereby causing inconveniences and damaging implications such as loss of finances, wrongful arrests, or serious psychological effects such as trauma. Figure 2.2 shows the basic working of the phishing process.

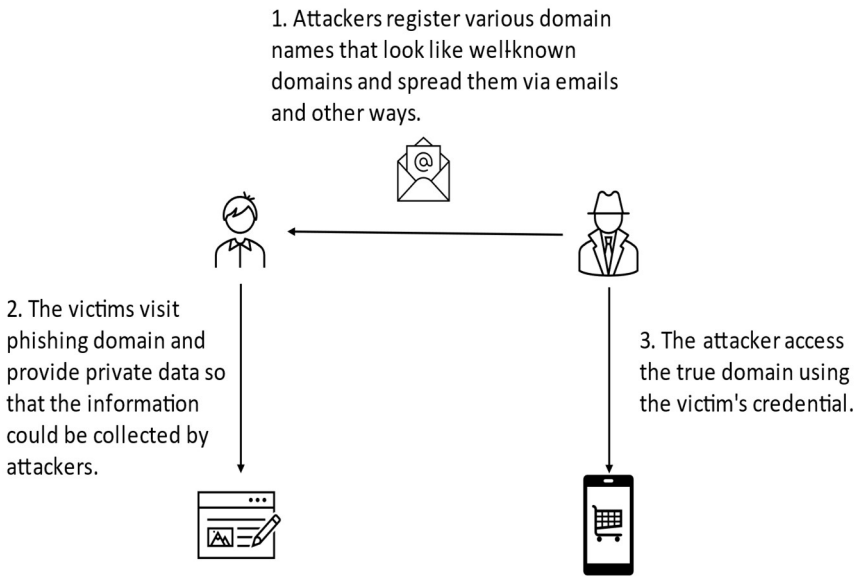

Fig 2.2 Phishing Process [53]

Legitimate domains: these are verified to be safe and have been carefully vetted by security experts. They are guaranteed to have all the properties of a secure and safe website such as secure certificates, valid WHOIS information, to name a few.

According to [53], malicious domains have various characteristics that are prevalent in phishing domains. Examples of such characteristics are dynamic IP, short lifespan, obfuscation, to name a few.

Phishing domains have become a nuisance because of the hardship that they impose on the victims, which include identity theft, financial loss or in some cases the loss of property.

In this project, the classification on datasets was done by assigning a Boolean value to each domain and adding a column named Malicious. All the URLs from the malicious (phishing) dataset had the malicious value of TRUE and the legitimate ones had the value FALSE.

Chapter 3: Architecture of the Detection System

3.1 Architecture and Processes

With the frequent updating of rules, pfSense Snort has the latest available rules for detecting the phishing. To maximize the chances of detection, Emerging Threats (ETs) open rules were considered [54]. These rules are distributed by Proofpoint, a security company that also distributes the rules for Snort and Suricata, another popular network IDS. After setting up the rules, I tested the Snort ability to detect the malicious domains. Because the focus of the project was on assessing the abilities of Snort, the pfSense gateway monitoring was disabled and only Snort, which is tweaked for different IPS policies (such as varying between connectivity, balanced, security and maximum detection), remained active on the firewall. It was also important to disable the blocking operation in order to minimize the disturbances during this test. The Snort performance in detecting the attacks was then evaluated and its performance measures were calculated.

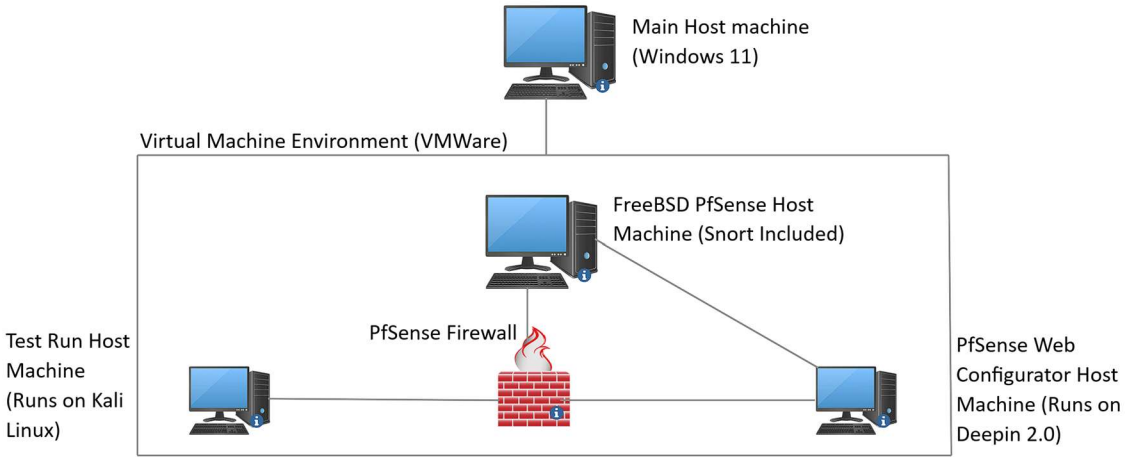

Fig 3.1 Architecture of the proposed detection system

Figure 3.1 illustrates the architecture of the proposed detection platform.

Snort was setup through pfSense by using a web configurator on a virtual machine running the latest version of Deepin, and VMWare was used to host the environment. The web configurator was accessed through the latest version of Google Chrome for the Debian platform. All the configurations made are applied to pfSense Snort, which was installed as a FreeBSD virtual machine instance in VMWare. It is important to give as many resources to this machine as possible since Snort is well known for being memory demanding.

The host machine was running on a core i5 13600k with 6 performance and 8 efficiency cores [56]. Each virtual machine had 8 GB of RAM and 8 core processors (by selecting 2 processors for each machine and 4 cores per processor). This was done to allow for a considerable performance speed while decreasing the chances of performance-related packet drops by Snort. The FreeBSD virtual machine had less storage space (roughly 15 GB) compared to the other virtual machines, each of which had 75 GB of storage space. This was mainly due to the sizes demanded by their respective compatible files. Due to the voluminous dataset, the best mechanism to run the detection solution was to do it through multithreading and multiprocessing due to reasons such as:

- Lack of proper hardware resources – running the program for making the requests for URLs one by one was time consuming and would have require about 2 weeks to complete, which was not ideal for keeping Wireshark running in the background, and so would face frequent crash.
- Long response times for some domains. To deal with this consistent issue, I had to enforce a timeout of 40 seconds.

Our implementation of Snort in pfSense had require the addition of the package through the package manager, followed by the configuration of the interface. Figure 3.2 shows a successfully installed and initiated instance of Snort for the Wide Area Network (WAN) interface. Additional configurations for any Snort interface could have been done using the pencil icon on the actions section under the Interface Settings Overview.

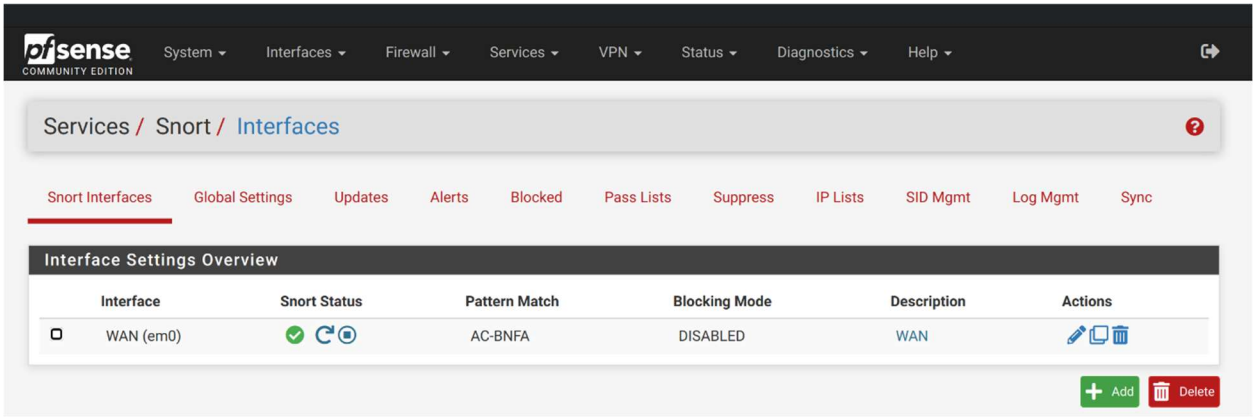

Fig 3.2 Snort interface in PfSense

With the successful installation and configuration of Snort, a customized configuration of the pfSense and its packages was imported, and an initial test run for verification was done.

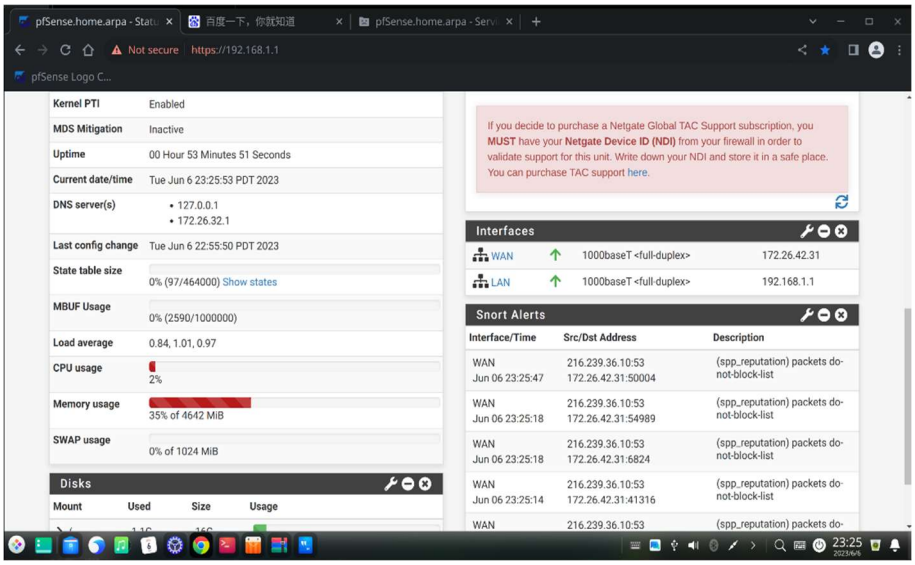

Fig 3.3 Configured pfSense with Snort in Deepin

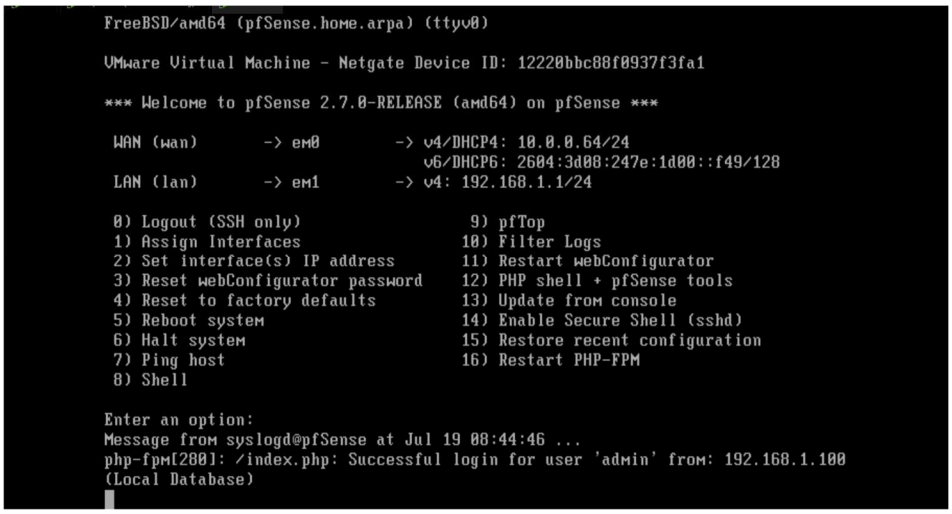

Fig 3.4 FreeBSD in working state.

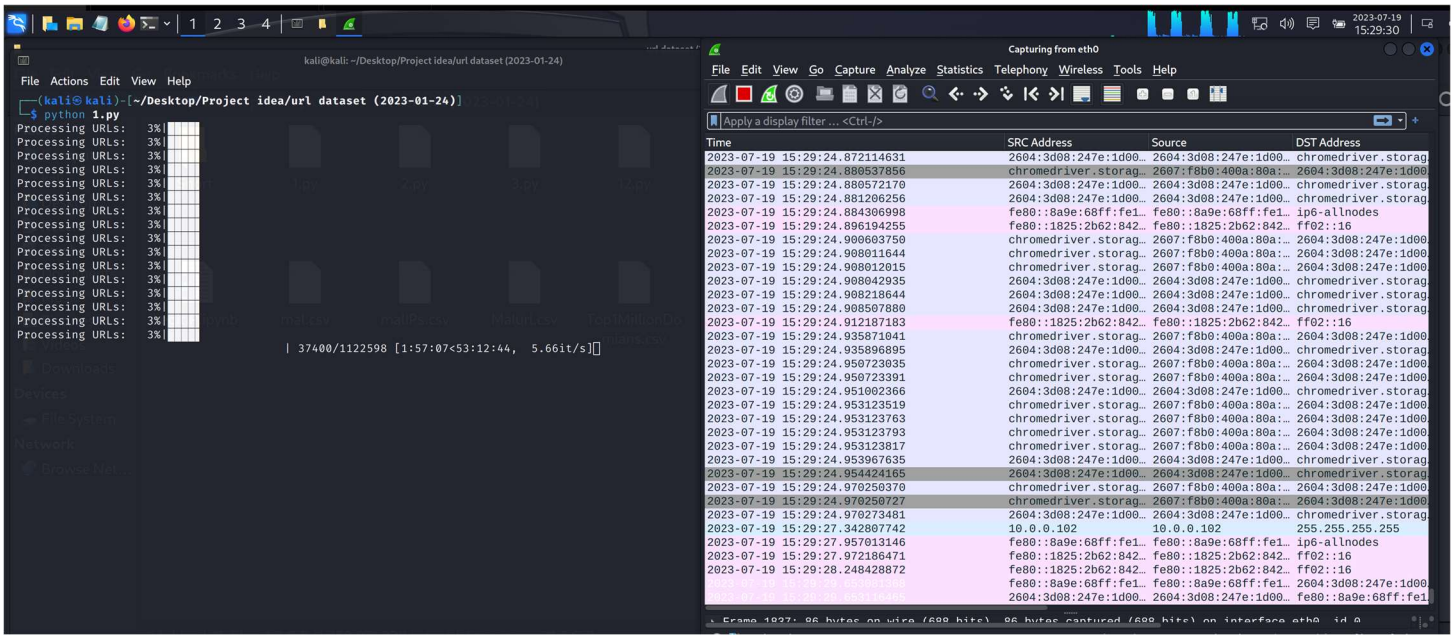

Fig 3.5 Kali Linux Virtual Machine running the Python script and Wireshark.

Figures 3.3 to 3.5 show three different virtual machines in working state. The platforms have their traffic controlled by the firewall pfSense, which in this case, has only been limited to Snort functionalities with no Gateway monitoring. This ensures that intrusion detection will happen with no interference from the firewall policies which may differ from those of Snort, and thus making pfSense a GUI terminal for solely network intrusion detection systems (NIDSs) management. In the Kali Linux VM, the python script opens a batch of 50 URLs in headless Chrome via the Selenium web driver per second in order to reduce the execution time as shown in Figure 3.5. The Deepin VM in Figure 3.3 is the pfSense web configurator machine for the FreeBSD VM (shown in Figure 3.4), which contains the firewall suite.

3.2 Data Extraction and Analysis.

Due to the dynamic nature of the domains' IP addresses and the Snort's inability to resolve the IP addresses (though pfSense provides a minimal functionality to do so one at a time), the only way to get the information needed to corroborate the URLs with Snort alerts was to use the Wireshark's packet file based on the requests done on the domains in the headless Chrome. The packet file was formatted by displaying the necessary columns which are Source (which contains the URL address from which the traffic originates), Source Address (which consists of the corresponding IP address for the source), Destination (which contains the URL address to which the traffic is received) and Destination Address (which consists of the corresponding IP address for the destination). After this, the packet file was exported as a CSV. The sources and their corresponding addresses were extracted into a new dataframe after which the destinations and their corresponding IP addresses were appended and all the duplicates were removed.

For Snort alerts, only the Source Address and Destination Address were extracted and appended in a single column dataframe. All the duplicates were also removed and the dataframe was saved.

All the above processes were done using database query techniques, in this case, duckdb for Python. The extracted Wireshark information were then used for assigning true addresses to the corresponding domains in the datasets, which were used with the extracted Snort alert addresses for evaluation purpose.

Chapter 4: Performance Evaluation

This Chapter describes the experimental evaluation processes as well as the analysis and comparison of results.

4.1 Datasets

In this study, two datasets of known URLs were collected, each one having either malicious or legitimate URLs, but not both. The first dataset contained legitimate links and it was obtained from the discontinued alexa.com repository containing one million URLs. However, a depreciated version of the dataset is still available for download [55], but it contains less domains than the final version. The second dataset contained malicious links and it was obtained from the COVID-19 Cyber Threat Coalition blocklist. The repository has been discontinued as well so that the vetting process that was used to check these links become unclear. The dataset for legitimate sites had 1 million URLs while the one for phishing domains had 122,596 URLs.

Both datasets were morphed into two separate CSV files by creating one column for the domains and one column named Malicious, the latter having a Boolean indicator on whether each URL is malicious or not for each of the files. During the test run of the domains, the two files were merged into one dataframe, which was then saved for later use.

4.2 Evaluation Metrics

To evaluate our proposed detection scheme, different metrics were considered, namely: accuracy, precision, recall, and false positive rate (FPR). The accuracy is defined as the ratio of correctly classified instances to the total number of instances. The precision is

defined as the ratio of correctly classified positive instances to the total number of instances classified as positive. The recall is defined as the ratio of correctly classified positive instances to the total number of positive instances, also known as the True Positive Rate (TPR). The FPR is defined as the ratio of legitimate links that are wrongly classified as malicious.

4.3 Dataset Attributes Criteria

The main attributes considered in this project were the URL from the domain, corresponding IP Address from the Wireshark dataset, the malicious status of the domains and the IP addresses from the Snort alerts. To derive the counts for each scenario on which the performance calculations were made. there was a matching of the corresponding IPs with those from the Snort alerts given each malicious status. I used the following notations for the key variables that were involved:

- df1: a single column dataframe from the Snort alerts file made by taking the Source Address and Destination Address and appending them one after the other.
- df2: a 2-column dataframe containing the domain and the respective malicious status value.
- df3: a 2-column dataframe made from extracting the Source and Destination domains which are appended to each other along with their respective IP addresses.
- merged_df: a 3-column dataframe obtained by performing a left outer join between df2 and df3 based on matching the domains in the two dataframes.

To understand the criteria used for the performance calculations, the following metrics were used for determining the confusion matrix:

- True Positive (TP): this is calculated by the count of all instances where an IP in merged_df matches one found in df1 and the corresponding malicious value is TRUE.
- False Positive (FP): this is calculated by the count of all instances where an IP in merged_df matches one found in df1 and the corresponding malicious value is FALSE.
- False negative (FN): this is calculated by the count of all instances where an IP in merged_df does not have a match in df1 and the corresponding malicious value is TRUE.
- True negative (TN): this is calculated by the count of all instances where an IP in merged_df does not have a match in df1 and the corresponding malicious value is FALSE.

4.4 Evaluation Procedure

The flowchart in Figure 4.1 illustrates the handling and processing of the data throughout the experiment and the complete structure. It shows how the experiment was carried out using a python program to run the headless Chrome considered the aforementioned two datasets whose traffic are monitored by both Snort and Wireshark running in different machines.

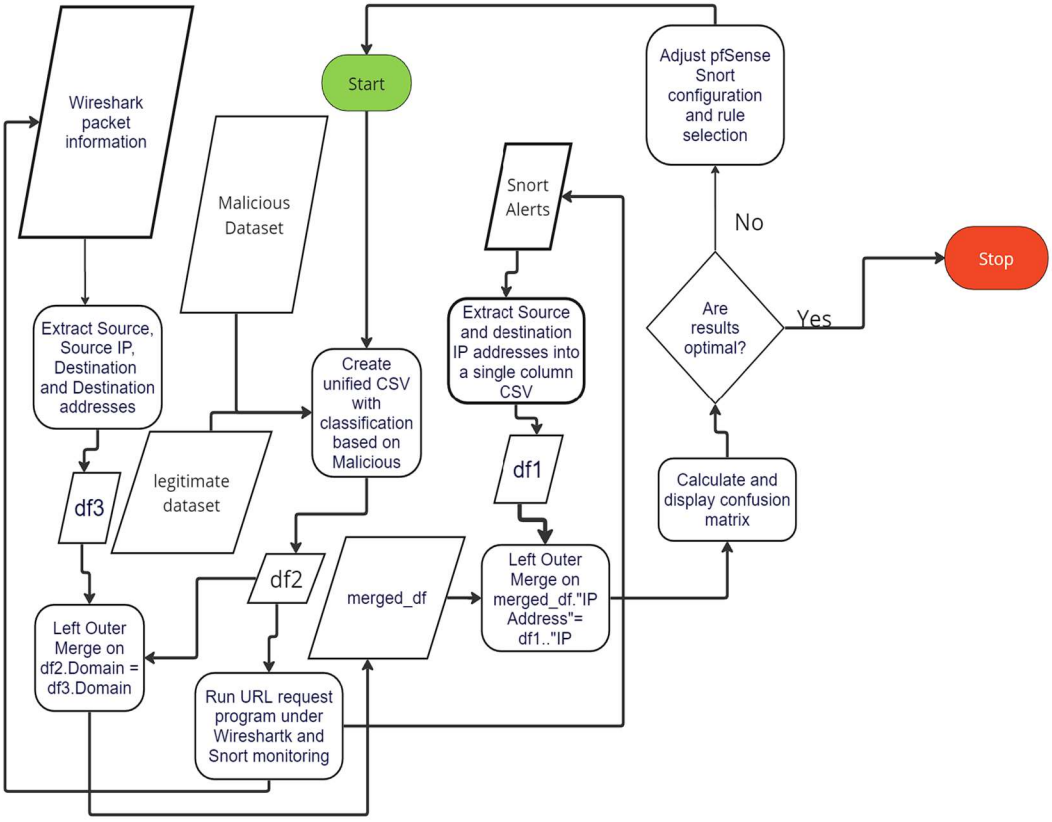

Fig 4.1 Flowchart of the experiment

In the experiment, the two domain datasets were merged, the malicious status was added, and the result was used for the headless Chrome requests under Snort and Wireshark monitoring. The Wireshark and Snort data were then processed as described in Section 3.2, after which the domain data was left merged with the Wireshark IPs. The resultant data were then utilized to check for matches of the IP address with the Snort alert IP addresses; and for each of the evaluation criteria scenarios for the confusion matrix, a count was recorded after which the confusion matrix was shown and the performance metrics were calculated.

4.5 Evaluation Results

Despite a few challenges, the proposed system managed to run and detect anomalies to a certain degree. The first issue was with the discontinuation of a significant number of domains, which not only shrunk the total number of URLs for testing to 588,766, but also the outcome of the evaluation performance. The duplicity in the Wireshark and Snort-detected traffic also heavily-influenced the outcome of the evaluation since some domains were sharing the same IP addresses.

Row Labels	Count of Msg
(http_inspect) INVALID CHUNK SIZE OR CHUNK SIZE FOLLOWED BY JUNK CHARACTERS	13233
ET POLICY DNS Query to DynDNS Domain *.serveirc .com	2338
(http_inspect) INVALID CONTENT-LENGTH OR CHUNK SIZE	1427
(http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE	911
ET TROJAN Possible Zeus GameOver/FluBot Related DGA NXDOMAIN Responses	674
ET POLICY DNS Query to DynDNS Domain *.servebeer .com	540
ET MALWARE All Numerical .ru Domain Lookup Likely Malware Related	500
(http_inspect) PROTOCOL-OTHER HTTP server response before client request	441
Phishing - Unicode Domain Spoofing	399
ET POLICY DNS Query to DynDNS Domain *.servehttp .com	332
ET POLICY DNS Query to DynDNS Domain *.serveftp .com	288
ET POLICY DNS Query to DynDNS Domain *.zapto .org	266
ET POLICY DNS Query to DynDNS Domain *.ddnsking .com	242
ET POLICY DNS Query to DynDNS Domain *.myvnc .com	238
ET TROJAN DNS Reply Sinkhole Microsoft NO-IP Domain	229
ET POLICY DNS Query to DynDNS Domain *.hopto .org	224
ET POLICY DNS Query to DynDNS Domain *.servehalflife .com	186
ET POLICY DNS Query to DynDNS Domain *.bounceme .net	182

Row Labels	Count of Classification
Potentially Bad Traffic	19035
Unknown Traffic	2842
A Network Trojan was Detected	1197
Misc activity	504
Potential Corporate Privacy Violation	118
Attempted Information Leak	16
Not Suspicious Traffic	1
(blank)	
Grand Total	**23713**

Fig 4.2 Alert count using a) Balanced and b) Security IPS policy modes.

As shown in Figure 4.2, the different IPS policies played a part in the obtained outcomes. However, the traffic captured in Wireshark showed some discrepancies between some of the IP addresses for the clean URLs and the malicious ones, yielding the possibility of false positives and false negatives. However, in the data pre-processing step, some actions were taken to eliminate the duplicates and rows with empty fields in the traffic data.

The considered metrics were calculated as follows:

- $Accuracy = \frac{TP + T}{TP + TN + FP + F}$

- $TPR = \frac{TP}{TP+}$
- $FPR = \frac{FP}{FP+TN}$
- $Recall = TPR$
- $Precision = \frac{TP}{FP+TP}$

With the above criteria, it would be extremely difficult to differentiate each IP address found in the Snort alerts. Because of the way pfSense Snort works, there was no clear and effective method to install the pre-processors which allow the users to view the alerts using resolved domain names instead of IP addresses without having to manually do it for each entry. The results that were obtained are shown in Figure 4.4

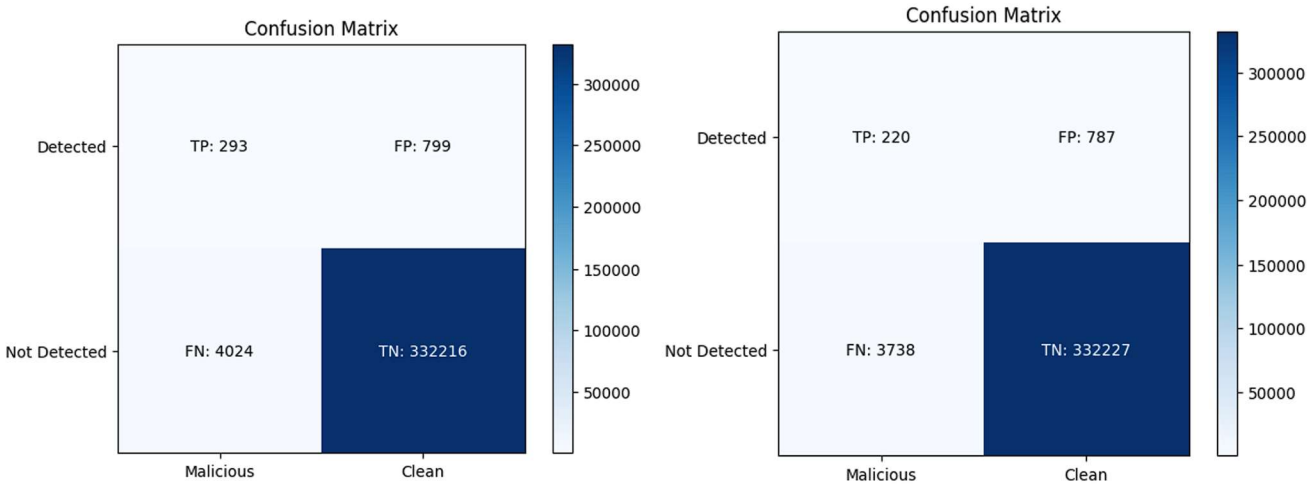

Fig 4.3 Confusion matrix for a) Balanced and b) Security IPS policy modes

In Figure 4.3, it is observed that there is a slight difference in the total number of samples. This is attributed to the fact that more packets were dropped in Balanced in order to

maintain consistent detection speeds as opposed to Security IPS which had a slower the response time.

The performance calculations were carried out in Google Colab for reliable speed in processing the large Wireshark and domain datasets. Table 4.1 shows the obtained performance measures. Despite achieving a remarkable 98% accuracy rate in both instances, the results came with mixed sentiments owing to the significant low True Positive Rates (TPR) of about 6% and 5% for Balanced and Security policies, respectively.

IPS Policy	**% TPR (Recall)**	**% FPR**	**% Accuracy**	**Precision%**
Balanced	6.78	0.24	98.6	26.8
Security	5.56	0.24	98.7	21.8

Table 4.1 Performance measures for Balanced and Security policies

The FPR values were low in both cases, which was a benefit, however, the TPR values were also low. This indicates that Snort could detect some malicious domains, but also would have missed many. The low precisions for both cases had demonstrated the low performance with the Security IPS mode having less efficiency compared to the Balanced policy one. This might likely be caused by the fact that more rules were loaded into the memory.

Given the results of this project, there are significant discrepancies in terms of existing rules and Snort capabilities as opposed to properly classifying the malicious domains before the attacks happen. Signature-based network IDPSs still have quite a journey in addressing the dynamic climate of cyber threats; thereby the outlook on IoCs must be

changed to address these challenges and focus on incorporating anomaly-based architectures, which will analyse the effect of online interactions on the host machines and networks on which they are connected.

4.6 Discussion

The implementation of this experiment was met with a couple of challenges. The Snort performance was significantly contradictory, with a high level of accuracy and an inversely low detection rate. The system's rate of detection was affected by the network speed, packet sizes [9], and the rate at which the domains were being opened. However, the system was prone to false positives, which could be due to its high sensitivity. The false positives could result in generating the unnecessary false alerts, which could lead to them being ignored or dropped.

To improve the Snort performance, the following suggestions prevail:

1. Reduce the false positives: the system sensitivity can be adjusted to reduce the false positives. This can be done by reducing the number of rules that are applied or increasing the threshold of the generated alerts and changing the toggling IPS policy to match the use case. This experiment could not go further due to the crashing of Snort in an attempt to use the Maximum detection policy as opposed to using the Balanced and Security modes. It should be noted that the latter is well known for increasing the chances of generating false positives.

2. Use of machine learning techniques: intrusion detection can be improved by using machine learning algorithms to identify the patterns of behaviour that are indicative

of an attack. This would help in creating more specific rules that can properly detect the anomalies [20].

3. Increasing the TPR: the TPR for this experiment was low. Increasing it would mean adding more rules that were disabled for this experiment, but that would mean some packets will likely be dropped, skipped, or ignored due to the Snort high-memory demand. This would demand for much more computing resources which was not available for this experiment.

4. Using Snort 3.0 could also be another progressive approach as it is deemed faster and better than Snort 2.x, which is the available version in the pfSense packages as of the period of the conducted experiment. Using the actual Snort software package along with an Apache-based web GUI such as Modsecurity [17], the main takeaway in this approach is the gimmicky nature of exporting and translating the configurations and the rules from the older Snort 2.x versions to their most recent counterparts. Another problem to be experienced would be that of having to go through the process of manually doing the configuration without a GUI (which makes it more difficult to track the changes made during the configuration) as the existing connector only works for the data extraction and processing phases.

Chapter 5: Conclusion and Future Work

5.1 Conclusion

Despite the overall high accuracy, our implementation fell short in correctly identifying a significant amount of phishing domains. That being said, more focus is needed to work on tweaks and improvement in the rule definitions as well as the performance robustness of pfSense Snort. No upgrade for the IDPS package on the firewall has been made available at the time of conducting this experiment, despite the fact that Snort has released newer iterations than the one that is available on pfSense.

5.2 Future Work

Given the limited number of resources for the conducted experiment, a starting point for improving the outcome of a GUI-based Snort will be having more powerful virtual machines by way of a high-end desktop machine at one's disposal. The availability of more resources can reduce the likelihood of performance inconsistencies in certain situations such as high-speed networks [20]. In addition, the use of more robust rules in paid subscriptions of ET Pro and the official Snort rule sets could be of great importance [4][5].

However, with the rise of attacks and new ways of gaining access to the victims' resources, some rules can be a little behind; thus, integrating our proposed system with repositories such as OWASP's ModSecurity Core Rule Set (CRS) [7] would be ideal as its policy will be based on OWASP's Top Ten threats [8]., which has the potential to greatly increase the accuracy, particularly the TPR since more well-defined rules will be in use. The authors in

paper [20] presented a new trajectory in the development of IDPSs using machine learning techniques and in the eventual process, integrating machine learning packages [21] into firewalls such as pfSense could have a positive impact on pfSense Snort. Another method to consider in rule management is the approach dealing with automatic rule generation [54], but using such approach is contingent on the availability of the necessary machine learning packages which are not presently available on pfSense.

True Positive Rate would need improvement by looking at new ways of writing rules that are directed at analysing domain traffic based on information from Wireshark packet traffic. The information can be utilised by analysing exported Wireshark packet CSV files and deducing an aggregate of attributes of malicious domains that differentiate them from legitimate ones through analytics and come up with key IOCs for the malicious websites that are specific to particular network traffic patterns.

Bibliography

[1] T. Li, “Stolen Canadian payment card info as cheap as fancy lattes,” IT World Canada – Information Technology News on Products, Services and Issues for CIOs, IT Managers and Network Admins, Jan. 2022. [Online]. Available: https://www.itworldcanada.com/article/stolen-canadian-payment-card-info-as-cheap-as-fancy-lattes/471522.

[2] N. Hildayanti and I. Riadi, “Forensics Analysis of Router On Computer Networks Using Live Forensics Method,” International Journal of Cyber-Security and Digital Forensics, vol. 8, no. 1, pp. 74-81, Jan. 2019. Doi: 10.17781/p002559.

[3] S. Dwiyatno, “Implementation of Snort IPS Using PfSense as Network Forensic in SMK XYZ,” 2019 International Conference on Information Management and Technology (ICIMTech), Jakarta, Indonesia, 2019, pp. 1-6. Doi: 10.1109/ICIMTech.2019.8843649.

[4] “ETPro versus ET Open Ruleset Comparison,” Proofpoint, Apr. 05, 2021. [Online]. Available: https://www.proofpoint.com/us/resources/data-sheets/etpro-versus-et-open-ruleset-comparison.

[5] “What are the differences in the rule sets?”, Snort FAQ, [Online]. Available: https://www.snort.org/faq/what-are-the-differences-in-the-rule-sets.

[6] H. Yan et al., “Cross-site scripting attack detection based on a modified convolution neural network”, Frontiers in Computational Neuroscience, vol. 16, 2022. Doi:10.3389/fncom.2022.981739.

[7] C.-H. Yang and C.-H. Shen, “IMPLEMENT WEB ATTACK DETECTION ENGINE WITH SNORT BY USING MODSECURITY CORE RULES,” ResearchGate, Apr. 2009, [Online]. Available: https://www.researchgate.net/publication/254110055_IMPLEMENT_WEB_ATTACK_DETECTION_ENGINE_WITH_SNORT_BY_USING_MODSECURITY_CORE_RULES

[8] “OWASP Top Ten”, OWASP Foundation. [Online]. Available: https://owasp.org/www-project-top-ten/.

[9] D. Zhang and S. Wang, “Optimization of traditional Snort Intrusion Detection System,” IOP Conference Series: Materials Science and Engineering, vol. 569, no. 4, p. 042041, 2019. Doi:10.1088/1757-899x/569/4/042041.

[10] A. Shrivastava, S. K. Choudhary and A. Kumar, “XSS vulnerability assessment and prevention in web application. 2016. Doi: 10.1109/ngct.2016.7877529.

[11] H. Alnabulsi, R. Islam, and Q. Mamun, “Detecting SQL injection attacks using SNORT IDS.” 2014. Doi: 10.1109/apwccse.2014.7053873.

[12] M. D. Azman, M. F. Marhusin, and R. Sulaiman, “Machine Learning-Based Technique to Detect SQL Injection Attack,” Journal of Computer Science, vol. 17, no. 3, pp. 296–303, Mar. 2021, doi: 10.3844/jcssp.2021.296.303.

[13] K. Patel and P. Sharma, “A Review paper on 28fSense – an Open source firewall introducing with different capabilities & customization,” International Journal of Advance Research and Innovative Ideas in Education,vol. 3, no. 2, pp. 635–641, Jan. 2017, [Online]. Available: http://ijariie.com/AdminUploadPdf/A_Review_paper_on_pfsense_–_an_Open_source_firewall_introducing_with_different_capabilities___customization_ijariie4108.pdf.

[14] M. Aggarwal, “Network Security with pfSense: Architect, deploy, and operate enterprise-grade firewalls.” Packt Publishing Ltd, 2018.

[15] N. Pais, “Cloud Firewall vs. Traditional Firewall – 5 Key Differences – 31West,” 31West:, Mar. 20, 2023. [Online]. Available: https://www.31west.net/blog/cloud-firewall-vs-traditional-firewall-5-key-differences/.

[16] R. U. Rehman, “Intrusion Detection Systems with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID.” Prentice Hall Professional, 2003.

[17] Akoul, “GitHub – akoul/ModSecurity-Snort: Snort connector for LibModSecurity (aka ModSecurity v3),” GitHub. [Online]. Available: https://github.com/akoul/ModSecurity-Snort.

[18] M. Roesch, “Snort – Lightweight Intrusion Detection for Networks.” 1999, pp. 229–238. [Online]. Available: http://ranger.uta.edu/~dliu/courses/cse6392-ids-spring2007/papers/USENIXLISA99-Snort.pdf.

[19] B. Caswell, J. Beale, and A. Baker, “Snort Intrusion Detection and Prevention Toolkit.” 2007. Doi: 10.1016/b978-1-59749-099-3.x5000-9.

[20] S. A. A. Shah and B. Issac, “Performance comparison of intrusion detection systems and application of machine learning to Snort system,” Future Generation Computer Systems, vol. 80, pp. 157–170, Mar. 2018, doi: 10.1016/j.future.2017.10.016.

[21] Q. A. Al-Haija and A. Ishtaiwi, “Machine Learning Based Model to Identify Firewall Decisions to Improve Cyber-Defense,” International Journal on Advanced Science, Engineering and Information Technology, vol. 11, no. 4, p. 1688, Aug. 2021, doi: 10.18517/ijaseit.11.4.14608.

[22] S. Dong, K. Abbas and R. Jain, “A Survey on Distributed Denial of Service (DdoS) Attacks in SDN and Cloud Computing Environments,” in IEEE Access, vol. 7, pp. 80813-80828, 2019, doi: 10.1109/ACCESS.2019.2922196.

[23] K. S. Bhosale, M. Nenova and G. Iliev, “The distributed denial of service attacks (DdoS) prevention mechanisms on application layer,” 2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS), Nis, Serbia, 2017, pp. 136-139, doi: 10.1109/TELSKS.2017.8246247.

[24] L. Wu, B. Brandt, X. Du and Bo Ji, “Analysis of clickjacking attacks and an effective defense scheme for Android devices,” 2016 IEEE Conference on Communications and NetworkSecurity (CNS), Philadelphia, PA, USA, 2016, pp. 55-63, doi: 10.1109/CNS.2016.7860470.

[25] B. Bhushan, G. Sahoo and A. K. Rai, “Man-in-the-middle attack in wireless and computer networking — A review,” 2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA) (Fall), Dehradun, India, 2017, pp. 1-6, doi: 10.1109/ICACCAF.2017.8344724.

[26] Y. Tao and K. Meng, “URL Based File Inclusion Attack Behavior Analysis and An Autoencoder Detection Model.” 2022. Doi: 10.1145/3532213.3532251.

[27] A. R. Baitha and S. Vinod, “Session Hijacking and Prevention Technique,” International Journal of Engineering & Technology, vol. 7, no. 2.6, p. 193, Mar. 2018, doi: 10.14419/ijet.v7i2.6.10566.

[28] S. Boztas, “Oblivious distributed guessing,” 2012 IEEE International Symposium on Information Theory Proceedings, Cambridge, MA, USA, 2012, pp. 2161-2165, doi: 10.1109/ISIT.2012.6283834.

[29] K. T. Dave, “Brute-force Attack ‘Seeking but Distressing,’” International Journal of Innovations in Engineering and Technology (IJIET), vol. 2, no. 3, Art. No. ISSN: 2319-1058, Jun. 2013.

[30] S. Gupta, A. Singhal and A. Kapoor, “A literature survey on social engineering attacks: Phishing attack,” 2016 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 2016, pp. 537-540, doi: 10.1109/CCAA.2016.7813778.

[31] D. Kshirsagar, S. Kumar and L. Purohit, “Exploring usage of ontology for HTTP response splitting attack,” 2015 1st International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 2015, , pp. 437-440, doi: 10.1109/NGCT.2015.7375156.

[32] B. Akour, “A Novel Approach for the Detection of DdoS Attacks using Snort,” 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), Paris, France, 2019, pp. 69-74, doi: 10.1109/CoDIT.2019.8820393.

[33] J. Wang, M. Lv and S. Cao, “A Study on Prevention and Defense System of DOS Attack Based on Snort,” 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2018, pp. 803-807, doi: 10.1109/IAEAC.2018.8570426.

[34] M. H. Bhuyan, D. K. Bhattacharyya and J. Kalita, “Network anomaly detection: Methods, systems and tools,” IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303-336, Firstquarter 2014, doi: 10.1109/SURV.2013.042313.00167.

[35] C. Biswas, P. Ghosh and R. Kar, “DoS/DdoS attacks and detection mechanisms: A survey,” Computers & Security, vol. 28, no. 6, pp. 236-250, Sep. 2009, doi: 10.1016/j.cose.2009.02.003.

[36] H. S. Kanhere and P. Mahalle, “A Review on Network Traffic Classification Using Machine Learning Techniques,” International Journal of Computer Applications, vol. 134, no. 4, pp. 1-6, January 2016, doi: 10.5120/ijca2016908242.

[37] Y. S. A. Younis, A. Omar and F. Aloul, "Intrusion detection system using machine learning techniques," 2014 9th International Conference on Informatics and Systems (INFOS), Cairo, Egypt, 2014, pp. 161-166, doi: 10.1109/INFOS.2014.6995399.

[38] A. R. Sharaf, E. S. Hafez, A. A. Khedr and H. A. Aly, "Optimizing Snort Performance Using Packet Bricks Framework," 2018 2nd International Conference on Information Science and System (ICISS), Cairo, Egypt, 2018, pp. 121-125, doi: 10.1109/ICISS.2018.8615904.

[39] R. S. Perdana, Y. W. Purwanto and M. Farid, "Comparative Performance Analysis of Snort as Intrusion Detection and Prevention System on Physical and Virtual Machines," 2020 1st International Conference on Computer Science and Information Technology (CSITech), Yogyakarta, Indonesia, 2020, pp. 1-6, doi: 10.1109/CSITech50342.2020.9266120.

[40] S. S. S. P. Pushpa Kumariand A. A. Mohamed, "A Study on Intrusion Detection and Prevention Systems in Computer Networks," 2017 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), Kannur, India, 2017, pp. 1-5, doi: 10.1109/ICCTICT.2017.7977627.

[41] H. Al-Abiad, A. Al-Yaseen and T. F. Mhamdi, "Performance Analysis of Intrusion Detection and Prevention System in High-Speed Networks," 2015 2nd International Conference on Computer and Information Sciences (ICCOINS), Kuala Lumpur, Malaysia, 2015, pp. 1-5, doi: 10.1109/ICCOINS.2015.7171794.

[42] A. Gupta, P. Rawat and S. Sharma, "A Comparative Study on the Performance Analysis of Snort IDS," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 425-430, doi: 10.1109/ICACCS48218.2020.9079773.

[43] A. H. Oluwafemi, A. F. Aibinu and J. A. S. Sanni, "Effectiveness of Snort IDS Performance for Network Security," 2015 International Conference on Computing, Communication and Security (ICCCS), Pamplemousses, Mauritius, 2015, pp. 1-5, doi: 10.1109/CCCS.2015.7398530.

[44] V. V. Dhanawade and R. R. Manthalkar, "Performance Analysis of Intrusion Detection and Prevention System Snort," 2017 International Conference on Data Management,

Analytics and Innovation (ICDMAI), Pune, India, 2017, pp. 375-380, doi: 10.1109/ICDMAI.2017.8073769.

[45] A. Shah, "Evaluation of Snort Intrusion Detection System Performance," 2019 International Conference on Smart Trends in Computing and Communications (SmartCom), Jaipur, India, 2019, pp. 592-596, doi: 10.1109/SmartCom48846.2019.8984871.

[46] A. K. Banerjee and J. Barik, "Performance analysis of snort on multi-core systems," 2012 Annual IEEE India Conference (INDICON), Kochi, India, 2012, pp. 506-511, doi: 10.1109/INDCON.2012.6420690.

[47] M. H. Kawsar, M. T. Rahman and M. G. H. Chowdhury, "Performance Analysis of Snort Intrusion Detection System," 2021 International Conference on Intelligent Computing and Communication (ICICC), Jaipur, India, 2021, pp. 1-5, doi: 10.1109/ICICC51527.2021.9443677.

[48] T. Patki, S. S. Gogte and A. Khare, "Performance Analysis of Intrusion Detection System: Snort vs Bro," 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, 2018, pp. 299-303, doi: 10.1109/GUCON.2018.8697215.

[49] R. Sharma and M. B. Chandak, "Performance analysis of Snort based Intrusion Detection System," 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 2017, pp. 699-704, doi: 10.1109/ICICCT1.2017.7974977.

[50] S. M. A. El-Tawab, "Performance analysis of Snort IDS for detecting intrusion attacks in Cloud Computing," 2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Monastir, Tunisia, 2017, pp. 326-332, doi: 10.1109/STA.2017.7939829.

[51] S. Kumar, R. Sehgal and J. S. Bhatia, "Hybrid honeypot framework for malware collection and analysis," 2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS), Chennai, India, 2012, pp. 1-5, doi: 10.1109/ICIInfS.2012.6304786.

[52] BackTrack Linux, "BackTrack Linux – Penetration Testing Distribution," BackTrack Linux. https://www.backtrack-linux.org/

[53] Bu, J. (2023). "Assessing the Effectiveness of Malicious Domain Prediction Using Machine Learning." A Report Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in the Department of Electrical and Computer Engineering, University of Victoria, 2023. [Online]. Available: http://hdl.handle.net/1828/15047. [Accessed: July, 05, 2023].

[54] "ET OPEN Ruleset Download Instructions." https://rules.emergingthreats.net/OPEN_download_instructions.html

[55] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip

[56] "How 13th Gen Intel® CoreTM Processors Work," Intel. https://www.intel.com/content/www/us/en/gaming/resources/how-hybrid-design-works.html