

NETWORK FORENSICS WITH MACHINE LEARNING

Introduction to Digital Forensics



ENSAY Mery;BODSON Fabrice - ENSAY Mery
HÖGSKOLAN I HALMSTAD 2023-2024

Table of Contents

Table of Figures.....	2
Abstract	3
Introduction.....	4
Related work.....	5
Contribution	6
Chapter 1 : Context Analysis.....	6
Chapter 2 : Analysis of existing	7
2.1 Tool comparisons	7
Chapter 3 : Requirements document	10
3.1 Add-on functionality	10
3.2 Report Example :	11
3.3 How the extension works.....	14
Chapter 4 : Create Model	15
Evaluation.....	17
Conclusion.....	19
Bibliography.....	20

Table of Figures

Figure 1 - Tools comparison	7
Figure 2 - Wireshark vs Snort	9
Figure 3 - Add-on fonctionnality	10
Figure 4 - Report example	11
Figure 5 - Use Case	14
Figure 6 - Execution result.....	17
Figure 7 - Results details	17

Abstract

This article presents a new approach to network forensics using machine learning.

The goal is to understand how machine learning can be used to analyze large numbers of network packets when a crime or incident has occurred, how to determine the origin of the attack or incident and identify patterns or intrusions.

To carry out this task, the paper first analyzes the context of network forensics and existing tools and techniques. It then defines the requirements of the ML model, including its functionality, the structure of the generated report, and how it will be integrated into an existing network analysis tool.

Finally, the paper evaluates the proposed ML model and compares it with existing works.

It concludes that ML can be used to improve the efficiency and accuracy of network forensics. The proposed ML model can identify patterns and intrusions in network traffic that humans would not be able to detect, much faster.

The paper also identifies some of the challenges of using ML in network forensics, such as the need for large datasets to train the model and the risk of bias in the model. However, the paper claims that the benefits of using ML outweigh the difficulties.

Overall, this paper presents a promising new approach to network forensics using ML. The proposed ML model can identify patterns and intrusions in network traffic that humans would not be able to detect, much faster.

Introduction

Network forensics is the process of collecting and analyzing raw network data and tracking network traffic to identify a cybercrime. It is an essential part of a digital investigation to understand what happened and to bring justice if a crime has been committed.

Machine Learning is a type of artificial intelligence that is capable to learn by itself, without being programmed. Machine Learning can be used to analyze a huge amount of data and to identify patterns that humans would not and in a faster way.

This paper presents a new approach to network forensics by using Machine Learning. Its objective is to understand how Machine Learning can be used to scan a huge amount of network packets when a crime or incident has occurred. The model should be able to determine the origin of the attack/incident, to identify patterns or intrusions.

To complete this task, this paper firstly focuses on the analysis of the context by precisely defining some aspects of this area. Then, we analyzed the existing to understand what tools exist for network analysis and how they work.

Thirdly, some requirements for the model must be defined like its functionality, the structure of the generated report and how this model will work. Finally, we explained how to integrate this model to an existing network analysis tool.

In the end, we evaluated our work in comparison to the existing work by testing it.

Related work

The network forensics field has a lot to gain by using the machine learning tools, in fact there has been a growing interest to it these past years.

The work of Chuwen Kuang [1] gives some interests in the area of anomaly detection with a framework that uses machine learning algorithms. He gives a detailed overview of the basic concepts of network forensics as well as basic classification methods for intrusion detection. He then proposes an intrusion detection model based on deep learning structure.

Another work worth to mention is provided by Roland Plaka [2] who proposes an intrusion detection system for a network and uses machine learning algorithms to classify network traffic as normal or malicious.

The use of machine learning would have as objective to identify malwares or reconstruct network attacks by analyzing the network packets. The effectiveness of network forensics can be improved by using machine learning, which is a powerful tool. It remains very crucial for machine learning algorithms to be effective, so they must be trained on substantial and representative data sets.

To operate this analysis, it is essential to use a network traffic analysis tool. Therefore, the work of Vivens Ndatinya [3] provides very useful and complete information about Wireshark, which is one of the most popular tool in this field, and how it can be used. Another interesting work is provided by Wonhyung Park and Seongjin Ahn [4] where very useful information can be found on Snort and Suricata.

Contribution

Chapter 1 : Context Analysis

This chapter will review the context and its various terms for a better understanding.

The idea of this project is to use Machine Learning (ML) to scan the packets on the network after an attack to analyze them and to understand the origin of it. The results of this analysis should be used to prevent future attacks and to make the ML model better.

First of all, the context concerns a specific branch of forensics science, namely "Network Forensic". What is Network Forensic ? It is the process of collecting and analyzing raw network data and tracking network traffic to investigate security incidents and identify intruders. By doing so, the goal is to prevent future intrusions and attacks (DDoS, DNS exfiltration, ...) by improving the defense mechanisms of the computer system.

Secondly, it is important to define what exactly is network packet capture. It is the process of using a tool to captures network packets that are flowing on a network with the objective to analyze them to get detailed information on the health of the network. By capturing these network packets, it is possible to dig deeper into specific packets get information about it like its source and destination, the ports number and eventually the data when the packet is not encrypted [5]. When a great number of packets are analyzed, it becomes possible to define traffic statistics and patterns.

A malicious packet can be identified by a known malicious domain, command-and-control communications, attack signatures or malicious web pages [6].

Chapter 2 : Analysis of existing

Before implementing the solution, we need to find out about the tools that already exist in the field. Today, many tools have been developed to help investigators in their network analysis. In fact, many proprietary tools already manage the network analysis with ML, like Fortinet's firewall. But these tools can be way too expensive and too powerful for the needs of some companies. Also, these tools are working in a live-time detection while our work aims to focus on a post-attack analysis.

Network forensic tools :

- **Tcpdump**[7] : is a command-line packet analyzer ; and libpcap, a portable C/C++ library for network traffic capture.
- **Tcpslice** [8]: is a tool for extracting portions of packet trace files generated.
- **Etherape** [9]: is network monitor for unix modeled after ethernan.
- **Netdude** [10](Network Dump data Displayer and Editor) : framework for inspection, analysis, and manipulation of tcpdump trace files
- **Argus** [11]: is an ongoing open-source network flow monitor project.
- **Wireshark** :[12] A network packet analyzer will try to capture network packets and tried o display that packet data as detailed as possible.
- **Snort / Suricate** [13]: is a network intrusion detection system that can be used to detect malicious activity on a network by examining network packets for patterns corresponding to known attacks or suspicious behavior.

2.1 Tool comparisons

Legend :

	Yes
	No

	Network traffic capture	Package analysis	Packet filtering	Header analysis	Intrusion detection	Identify devices
Tcpdump						
Tcpslice						
Etherape						
Netdude						
Argus						
Wireshark						
Snort						
Suricate						

Figure 1 - Tools comparison

Each tool includes functions for capturing, analyzing and filtering packets and analyzing headers. By analyzing the various existing tools, we discovered that several open-source tools exist, which makes it possible to retrieve requests on a network in real time, but after an attack they do not give an automatic report on what caused the problem.

The solution to the problem would be to develop a complementary tool to those that already exist, which could be launched after an attack to automatically analyze packets and, thanks to machine learning, highlight the packets that caused the problem in order to protect against them in the future.

In order to achieve this, we decided to focus on Wireshark [3] and on Snort [4] as they are both very popular tools in the network packets analysis domain and they both are open source and free to use. To have the best comparison out of them and to select the one we will focus on, we have established a few questions and we have provided the answers in a table :

	Wireshark [14]	Snort [15]
Is it possible to add an extension to the tool?	Yes, because the tool is open-source and free. So, it is very easy and very common to adapt the source code to your needs.	
Is it possible for the tool to read a file? What type of file?	Pcap, Pcapng, Libpcap, Snoop, Atmsnoop, Microsoft Network Monitor, Novell LANalyzer, AIX iptrace, Cinco Networks NetXray	
Does the tool have the functionality to malicious activity detection?	Yes, by applying filters to registered packets	Yes, in "log" mode, Snort doesn't capture network traffic in real time, but it analyzes network logs that have already been recorded.
What are the tool's limitations?	<ul style="list-style-type: none"> - Cannot detect all network attacks. - Cannot provide information on the attacker's intentions. - Can be complex to use. - Cannot detect attacks that use encryption mechanisms. 	<ul style="list-style-type: none"> - Cannot detect all network attacks. - May generate false positives. - Can be complex to configure and use. - Cannot detect attacks that use encryption mechanisms. - Cannot detect attacks hidden in

	<ul style="list-style-type: none"> - Cannot detect attacks based on unknown vulnerabilities. 	legitimate network traffic. <ul style="list-style-type: none"> - Cannot detect attacks based on unknown vulnerabilities.
--	---	---

Figure 2 - Wireshark vs Snort

Based on this comparative table, we chose to select Wireshark as this tool has less limitations.

Chapter 3 : Requirements document

Following the analysis of the existing system, one possible solution is to develop an extension using machine learning to detect attacks on the network.


This will make it possible to analyze network packets retrieved in the past. The model doesn't work in real time, it's only if the user uses it by giving it a network traffic record, the extension will provide a detailed report of what it has analyzed.

3.1 Add-on functionality

Features	Description
Package recovery	The user will provide the network traffic logs he made with WireShark on the day of the problem.
Package analysis	Each packet is analyzed to detect whether it is part of the attack or malicious.
Attack detection	Once each packet has been analyzed, the model will be able to recognize the type of attack (DDOS, Zero-day, Phishing, etc.).
Report Generation	Once the extension has received the results of the model, with the packets causing the problem, the type of attack, ... a report will be generated

Figure 3 - Add-on fonctionnality

3.2 Report Example :



Network analysis report

Date of analysis : Number of analysis :

Objectif :

-
-
-

Problematic package :

	Number	IP Source	IP Destination	Protocol	Information
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

Type of attack detected :

Recommendation :

.....

.....

Figure 4 - Report example

Explanation of sections :

- The "Problematic package" section is a table containing information about each packet that has been recognized as malicious by the model or pheasant.
- The "Type of attack detected" section contains information about the attack detected, such as its type, name, etc.
- The last section is "Recommendation", which contains all the information you need to protect yourself against the attack detected in the future.

But to set up such an extension, certain questions need to be asked.

1. How to add an extension to Wireshark ? (Criteria, ...) [16]

First step, create your extension. You can either create a Lua script or choose your language and compile it to machine code.

Second step, place your extension in the Wireshark plugin directory. The plugin directory location depends on your operating system:

- Windows: %APPDATA%\Wireshark\plugins
- MacOS: /Library/Application Support/Wireshark/plugins
- Linux: /usr/share/wireshark/plugins

Last step, restart Wireshark and enable your extension (Edit > Preferences > Plugins. Select your extension from the list and click). [16]

2. What format will the model take the data in? Is it possible to use wireshark's basic format?

Wireshark supports several different capture file formats. Its default format is "pcapng". It is very flexible but other tools may not support it. The problems with other formats are that they are not extensible and lack some information that would be really helpful. [17]

Is it possible to use the format for machine learning? "pcapng" files can be read with python. [18] [19] This enables pre-processing to be carried out so that the data supplied to the model is easy to use.

3. How can the model be taught to detect a malicious package? (criteria, ...) And how can you recognize an attack?

An intrusion detection system (IDS) is a mechanism designed to detect abnormal or suspicious activity on the target being analyzed (a network or a host). It provides information on both successful and failed intrusion attempts. The best-known IDSs are signature-based detection and anomaly-based detection. [20]

Signature-based detection works using a pre-programmed list of known threats and their indicators of compromise. Its disadvantages are that it can only recognize known attacks and requires daily updates to keep abreast of new attacks. This system is easily circumvented by hackers. They use so-called "evasion" techniques, which involves disguising the attacks used. These disguise techniques tend to vary the attack signatures, so that they are no longer recognized by the IDS. [21]

Signature example :

- Buffer overflow [22] : Buffer overflows usually contain shellcodes. Simply keep a list of current shellcodes and check whether a request contains a shellcode. This is the method used by IDS, but the model needs to learn how to identify shellcodes.
- DoS/DDoS [15] : which are mainly malformed packets and protocol attacks. These attacks aim to render a service inaccessible by saturating it with traffic. Signatures of DoS/DDoS attacks can identify sudden spikes in traffic from a single source.
- DNS [23]: There are many different types of DNS exfiltration attack, each with its own signature. Here are some examples of DNS exfiltration attack signatures:
 - Unusual DNS queries : such as queries for unknown domains or queries for sensitive data, can be an indication of a DNS exfiltration attack.
 - Unusual DNS query strings : such as queries for domains that are not directly related to each other, can be an indication of a DNS exfiltration attack.

Anomaly-based detection consists of looking for events that do not correspond to the normal behavior of a system.

Example with DNS [24]:

- The approach is to look for what you'd expect to see, but which isn't there. A DNS query is only performed before another query such as http. The detection method consists of looking for DNS requests that do not have a corresponding request by another application such as http. There will be exceptions which can easily be filtered out.

3.3 How the extension works

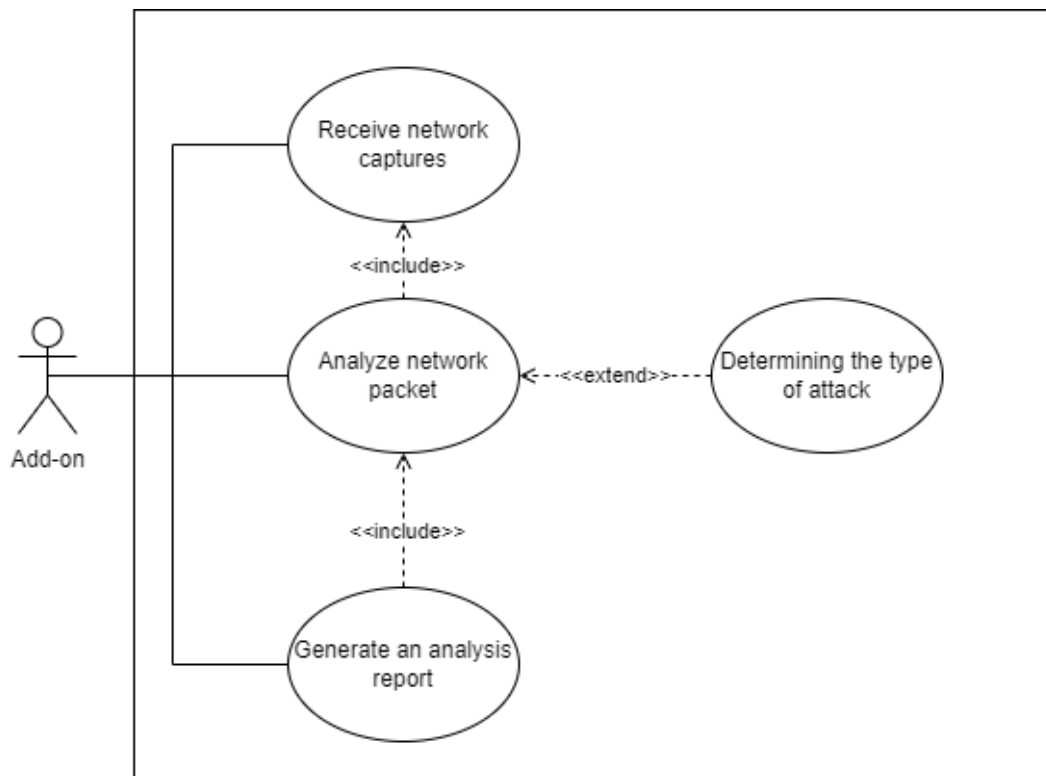


Figure 5 - Use Case

In the use-case above, a presentation of the various functions that the extensions will be able to perform. The extension will be added to Wireshark as explained above.

It allows the user to import a file containing a Traffic record from a network. This import will be used to launch an analysis of the various packets, using a machine learning model to detect potential attacks based on packet signatures or anomalies.

Once the analysis of the recording is complete, the model will determine the type of attack the network has suffered and send the user a detailed report.

Chapter 4 : Create Model

1. Collect and clean learning data

First step is to find the dataset:

<https://www.kaggle.com/datasets/cicdataset/cicids2017/data>

Second step is to clean the dataset:

- Delete incomplete or corrupt rows. This includes rows with missing values or outliers.
- Convert all values to the same data type. This will make it easier for machine learning algorithms to use the dataset.
- Normalise the data. This means scaling all the data so that it is in the same range. This prevents certain features from dominating others.

2. Choose a machine learning model.

Here, you need to know how to classify whether the package is "BENIGN" or the type of attack. So there are several classes. To choose the right model, you need to analyse the data and see if it has any particular criteria. Not being experts in machine learning algorithms, the one that seemed easiest to us was the one based on decision trees.

3. Train the machine learning model

With the data that has been recovered, the model can be trained and evaluated.

```
# Load the dataset
df = pd.read_csv("./MachineLearningCVE/Thursday-WorkingHours-Afternoon-
Infiltration.pcap_ISCX.csv")
#Delete rows with null values
df = df.dropna()

# Créez un encodeur pour la colonne "Label"
encoder = LabelEncoder()
df[" Label"] = encoder.fit_transform(df[" Label"])

# Convert all values to the same data type
df = df.astype(np.float32)
df = df[np.isfinite(df).all(axis=1)]
```



```
# Separate the dataset into two sets: a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(df.drop(" Label",
axis=1), df[" Label"], test_size=0.25)

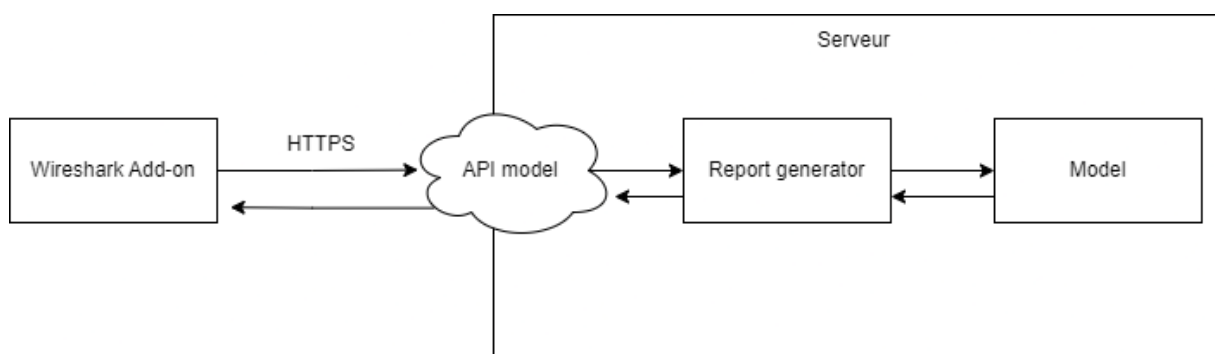
# Train a machine learning model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Evaluate the model's performance
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
confusion_matrix = confusion_matrix(y_test, y_pred)
```

Here, the example of a machine learning model is quite simple. If our knowledge in the field were more extensive, we could have used a neural network to make the model more powerful.

The implemented example only uses the various packages themselves, so it relies on their signature. It should be improved to enable it to recognize anomalies as well.

4. Integrate the machine learning model into Wireshark.



To make the add-on functional in Wireshark, you'll need to set up a server with an API that will allow the extension to contact it. On the server, 2 modules will be launched, the first to generate the report, the second to launch the traffic analysis model. The API will return the report to the extension, which can then be downloaded by the user.

5. Test the Wireshark extension.

Evaluation

Once our program has been coded, we experimented it and we can see that 72.099 packets have been captured and analyzed.

On the following figure, the model has analyzed the packets, and it results in 72.093 BENIGN packets and 3 were an attack. There are also 3 other packets that were wrongly classified as Infiltration.

```
Entrée [13]: print(confusion_matrix)
              [[72093    0]
               [    3    3]]

Entrée [14]: encoder.classes_
Out[14]: array(['BENIGN', 'Infiltration'], dtype=object)
```

Figure 6 - Execution result

```
Entrée [9]: # Évaluez les performances du modèle
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)

Accuracy: 0.9999583905463322
Precision: 1.0
Recall: 0.5
F1-score: 0.6666666666666666
```

Figure 7 - Results details

Nowadays papers are already showing the use of Machine Learning for network forensics in many ways and with different objectives at the end. Our work focused on developing and adding an extension to a powerful network analysis tool such as Wireshark so it can be possible to use this popular tool with a machine learning capable of classifying packets and to generate a report.

As explained previously in this paper, now that the extension exists, it must be placed in the Wireshark plugin directory. The plugin directory location depends on the operating system:

- Windows: %APPDATA%\Wireshark\plugins

- **MacOs:** `/Library/Application Support/Wireshark/plugins`
- **Linux:** `/usr/share/wireshark/plugins`

Last step, Wireshark should be restarted, and the extension enabled. Now it is possible to use the extension on Wireshark.

Most of the recent research focuses on the development of a machine learning model capable of this kind of analysis as we can read in Roland Plaka's [2] and Chuwen Kuang's [1] works. While these papers present a good analysis of models, classification etc. they do not offer an easy way to use their model, not as easy as this paper does.

Conclusion

Nowadays papers are already showing the use of Machine Learning for network forensics in many ways and with different objectives at the end.

Our work focused on developing and adding an extension to a powerful network analysis tool such as Wireshark so it can be possible to use this popular tool with a machine learning capable of classifying packets and to generate a report.

To complete this task, this paper firstly presented the analysis of the context by precisely defining some aspects of this area. Then, we analyzed the existing to understand what tools exist for network analysis and how they work to be capable to use them correctly.

Thirdly, some requirements for the model have been defined like its functionality, the structure of the generated report and how this model should work. Finally, we explained how to integrate this model to an existing network analysis tool like Wireshark that we selected based on several criteria.

In the end, we evaluated our work by testing it and by analyzing the results. We concluded by explaining how our work is different from what already exists.

Bibliography

- [1] C. Kuang, 'Research on Network Traffic Anomaly Detection Method Based on Deep Learning', *J. Phys.: Conf. Ser.*, vol. 1861, no. 1, p. 012007, Mar. 2021, doi: 10.1088/1742-6596/1861/1/012007.
- [2] R. Plaka, 'INTRUSION DETECTION USING MACHINE LEARNING FOR INDUSTRIAL CONTROL SYSTEMS'.
- [3] V. Ndatinya, Z. Xiao, V. Manepalli, K. Meng, and Y. Xiao, 'Network forensics analysis using Wireshark', *International Journal of Security and Networks*, vol. 10, p. 91, Jan. 2015, doi: 10.1504/IJSN.2015.070421.
- [4] W. Park and S. Ahn, 'Performance Comparison and Detection Analysis in Snort and Suricata Environment', *Wireless Pers Commun*, vol. 94, no. 2, pp. 241–252, May 2017, doi: 10.1007/s11277-016-3209-9.
- [5] L. Deri, 'Improving Passive Packet Capture: Beyond Device Polling'.
- [6] T. Ongun, T. Sakharov, S. Boboila, A. Oprea, and T. Eliassi-Rad, 'On Designing Machine Learning Models for Malicious Network Traffic Classification'. arXiv, Jul. 10, 2019. Accessed: Sep. 25, 2023. [Online]. Available: <http://arxiv.org/abs/1907.04846>
- [7] 'Home | TCPDUMP & LIBPCAP'. Accessed: Sep. 24, 2023. [Online]. Available: <https://www.tcpdump.org/>
- [8] 'tcpdump(8) - Linux man page'. Accessed: Sep. 24, 2023. [Online]. Available: <https://linux.die.net/man/8/tcpdump>
- [9] 'EtherApe, a graphical network monitor'. Accessed: Sep. 24, 2023. [Online]. Available: <https://etherape.sourceforge.io/>
- [10] 'Netdude'. Accessed: Sep. 24, 2023. [Online]. Available: <https://netdude.sourceforge.net/>
- [11] 'Argus – Audit Record Generation and Utilization System', *Wikipedia*. Sep. 04, 2023. Accessed: Sep. 24, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Argus_%E2%80%93_Audit_Record_Generation_and_Utilization_System&oldid=1173843773
- [12] 'Wireshark Developer's Guide'. Accessed: Sep. 29, 2023. [Online]. Available: https://www.wireshark.org/docs/wsdg_html_chunked/
- [13] 'Snort - Network Intrusion Detection & Prevention System'. Accessed: Sep. 24, 2023. [Online]. Available: <https://www.snort.org/>
- [14] '1.4. Development And Maintenance Of Wireshark'. Accessed: Sep. 26, 2023. [Online]. Available: https://www.wireshark.org/docs/wsdg_html_chunked/ChIntroDevelopment.html
- [15] 'SNORT Users Manual 2.9.16'. Accessed: Sep. 26, 2023. [Online]. Available: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/>
- [16] 'Wireshark Developer's Guide'. Accessed: Oct. 04, 2023. [Online]. Available: https://www.wireshark.org/docs/wsdg_html_chunked/

- [17] ‘Appendix B. Files and Folders’. Accessed: Oct. 04, 2023. [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/AppFiles.html
- [18] S. Santi, ‘Python-pcapng’. Sep. 06, 2023. Accessed: Oct. 04, 2023. [Online]. Available: <https://github.com/rshk/python-pcapng>
- [19] ‘python-pcapng/pcapng-docs/PCAP Next Generation Dump File Format.html at master · rshk/python-pcapng’, GitHub. Accessed: Oct. 04, 2023. [Online]. Available: <https://github.com/rshk/python-pcapng/blob/master/pcapng-docs/PCAP%20Next%20Generation%20Dump%20File%20Format.html>
- [20] ‘What is an intrusion detection system (IDS)? | IBM’. Accessed: Oct. 04, 2023. [Online]. Available: <https://www.ibm.com/topics/intrusion-detection-system>
- [21] ‘Les systèmes de détection d’intrusion’. Accessed: Oct. 04, 2023. [Online]. Available: https://www.securiteinfo.com/conseils-cybersecurite/choix_ids.shtml
- [22] ‘What is signature-based detection?’ Accessed: Oct. 04, 2023. [Online]. Available: <https://www.educative.io/answers/what-is-signature-based-detection>
- [23] ‘34152.pdf on Egnyte’, Egnyte. Accessed: Oct. 04, 2023. [Online]. Available: <https://sansorg.egnyte.com/dl/r4ouqZy5dp>
- [24] M. Shurman, R. Khrais, and A. Yateem, ‘DoS and DDoS Attack Detection Using Deep Learning and IDS’, *IAJIT*, vol. 17, no. 4A, pp. 655–661, Jul. 2020, doi: 10.34028/iajit/17/4A/10.