HOST-BASED INTRUSION DETECTION SYSTEM

FOR SMALL NETWORKS

by

Syed Sohaib Shah

Hamid Reza

Shayan Ahmad

Supervisor

Sir Faran Mehmood

Department of Computer Science

Institute of Space Technology, Islamabad

July 2024

HOST -BASED INTRUSION DETECTION SYSTEM

FOR SMALL NETWORKS

by

Syed Sohaib Shah

Hamid Reza

Shayan Ahmad

Supervisor

Sir Faran Mehmood

APPROVAL BY BOARD OF EXAMINERS

______________________ ______________________

Supervisor Co-Supervisor

______________________ ______________________

Internal Examiner External Examiner

AUTHORS DECLARATION

I take full responsibility for the research work conducted during the thesis titled "**Host-Based Intrusion Detection System for Small Networks**".

I solemnly declare that the research and development work presented in the thesis is done solemnly by me with no significant help from any other person; however, small help wherever taken is duly acknowledged. I have also written the thesis by myself. Moreover, I have not presented this thesis or any part of the thesis previously to any other degree awarding institution within Pakistan or abroad.

I understand that the management of IST has a zero-tolerance policy towards plagiarism. Therefore, I as an author of the above-mentioned thesis solemnly declare that no portion of my thesis has been plagiarized and any material used in this Thesis does not contain any literal citing (verbatim) of more than 70 words (total) even by giving a reference unless I have obtained the written permission of the publisher to do so. Furthermore, the work presented in the thesis is my own original work and I have positively cited the related work of the other researchers by clearly differentiating my work from their relevant work.

I further understand that if I am found guilty of any form of plagiarism in my thesis work even after my graduation, the institute reserves the right to revoke my BS degree. Moreover, the Institute will also have the right to publish my name on its website that keeps a record of the students who plagiarized in their thesis work.

Syed Sohaib Shah

Hamid Reza

Shayan Ahmad

I hereby acknowledge that submitted thesis is final version and should be scrutinized for plagiarism as per IST policy

Sir Faran Mehmood

Dated: July 8, 2024

____

Verified by Plagiarism Cell Officer

Dated______________

CERTIFICATE

This is to certify that the research work described in this thesis is the original work of the author and has been carried out under my direct supervision. I have personally gone through all the data/results/materials reported in the manuscript and certify their correctness/authenticity. I further certify that the material included in this thesis is not plagiarized and has not been used in part or full in a manuscript already submitted or in the process of submission in partial/complete fulfillment of the award of any other degree from any institution. I also certify that the thesis has been prepared under my supervision according to the prescribed format and I endorse its evaluation for the award of Bachelor of Science in Electrical Engineering degree through the official procedures of the Institute.

Sir Faran Mehmood

DEDICATION

This research work is dedicated to our beloved Parents and Teachers and Friends who supported and motivated us in tough times and made us into what we are today.

ABSTRACT

This paper provides an extensive explanation of the development and implementation of a Host-Based Intrusion Detection System (HIDS) which focuses on enhancing system security via different steps and measures. It includes “Manual and Automatic Port Filtering”, “Packet sniffing”, “Comparing against YARA Rules”, and “ML for the detection of Malicious EXE files”. One of the main purposes of the system is to block ports that are to be considered as not necessary for our organization and by doing so we will eventually mitigate any threat from those ports. Securing those ports can tackle several security threats which why we will monitor network traffic and in real time test against YARA rules. If the traffic is HTTPS, then there is going to be some encryption so we will keep monitoring our download folder and our ML algorithm will try and see if there is any EXE file there or not if it detects an EXE file it extracts the PE information and then inform the user if that file is malicious or not.

Table of Contents

1 INTRODUCTION 1

2 implementation 1

2.1 IP Blocking 1

2.2 Port Filtering 2

2.3 Packet Sniffer 2

2.4 Yara Rules 3

2.5 AI/ML 4

3 Workflow 5

4 Background Information 7

5 Related Work 8

6 Results and Discussions 9

7 Conclusion and Future Work 10

7.1 Conclusion 10

7.2 Future Work 10

8 References 11

LIST OF FIGURES

Figure 1 Example of YARA Rules 3

Figure 2 Features of PE Files 4

LIST OF ABBREVIATIONS AND SYMBOLS

HIDS	Host-Based Intrusion Detection System
YARA Rules	Yet Another Recursive Acronym
EXE Files	Executable Files
PE File	Portable Executable File
HTTPS	Hyper Text Transfer Protocol Secure
NIDS	Network-Based Intrusion Detection System
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
DDoS	Distributed Denial of Service
FTP	File Transfer Protocol
OSI Model	Open Systems Interconnections Model

1 INTRODUCTION

In this age of Modernization and Information Technology, the use of the internet is quite widespread. Every person is at least somewhat has been exposed to the Internet. The Internet is home to billions of computers and other electronic devices. The easy availability of the Internet to the public has placed quite a strain on security concerns. With that, the need for cyber security, to provide protection against outside threats has been more than ever. There are numerous technologies and methods to safeguard one's system/device from being exposed to vulnerable software/files. There are two ways to detect Intrusion on one's system. The first one is Network Based Intrusion Detection System, and the second one is Host Based Intrusion Detection System. Intrusion Prevention systems are used to mitigate intrusions which are beyond the scope of our project. The initial type of IDS was called HIDS (Host Based IDS), and its primary application was for internal monitoring (inside a computer or other device). However, more recently, several HIDS variations that may be utilized for network monitoring have been developed. [1] Host-Based Intrusion Detection Systems (HIDS) are crucial when we are attempting to secure individual systems from unauthorized access and misuse of critical data. Different from network-based intrusion detection systems (NIDS), which keep track of the network traffic for suspicious activity, HIDS prioritizes the monitoring and analysis of the events occurring inside a certain host/system. This gives us additional knowledge and control over the security and maintenance of the system. There is one other kind of IDS known as Hybrid Intrusion Detection, which combines host-based and network-based intrusion detection to give a system greater security and dynamics. [1] This paper outlines the architecture, design, and implementation of a HIDS that improves and enhances the security of the system through different techniques such as port filtering, monitoring network traffic and Machine Learning.

2 IMPLEMENTATION

2.1 IP Blocking

IP blocking in HIDS is, therefore, one of the key steps toward the enforcement of better security in a small network, which averts potential threats by preventing unauthorized access to the network. At the host level, the HIDS will be continuously watching all incoming and outgoing traffic in the small network environment. The system can hence automatically block the IPs of interest in those cases where the suspicion arises or a pattern typical of malware behavior is noted. It helps in stopping attacks, such as brute force attempts, port scanning, and unauthorized access, among others, before they get a chance to break into a system. In this regard, IP blocking can help small networks achieve a secure environment where only legitimate traffic is allowed, hence reducing the potential risk of data breaches among other cyber threats. This provides quite an effective defense measure with little infrastructure required for small networks.

2.2 Port Filtering

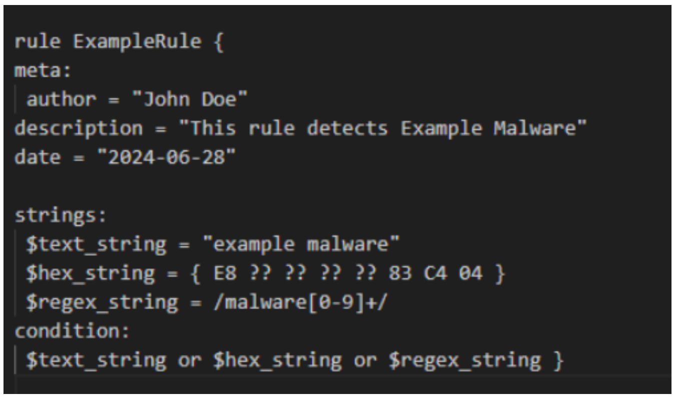
A port is a virtual endpoint that helps the computers to differentiate between different network traffic. Each port has a number assigned to it called a” Port Number”. It is a 16-bit unsigned integer that ranges from 0 to 65535. [2] They are managed by OSI. On a 7-layer OSI model, the concept of Port is to be found on the fourth layer. The fourth layer is the transport layer. TCP and UDP on the transport layer indicate which port should the packet be assigned to. Each port has its own designated service that they carry out and perform. [2] Port 0 to 1023 are referred to as “well-known” ports. There are two states of a port. Either a port is open or closed. An Open Port refers to a TCP or UDP port number that is set to accept packets. A Closed Port is a port that rejects connections or ignores all packets that it receives. Each port has its vulnerability. A port vulnerability is a weakness or flaw in a network port, which can be exploited by attackers to gain unauthorized access to systems or data. For instance,” Port 135” if left open may allow the attackers to execute arbitrary code, gain unauthorized access to sensitive data or launch DDoS attacks. Port 21 used by FTP, File Transfer Protocol. Unrestricted inbound access to this port can allow attackers to connect to the FTP server and potentially exploit vulnerabilities, leading to unauthorized access or data ex filtration. To mitigate Port vulnerability, we must apply port filtering. When we restrict a port from communicating online it is called port filtering. The ports filtered are categorized as filtered ports. We selected a list of common ports and restricted the rest of the ports. Common ports include port 80, port 443, port 22 and so on. There is also a feature included via which one can block/unblock any port. The study indicates that though a few open ports remain in most desktop-based user computers, it clarifies that systems can be made tight by ensuring that no unnecessary ports are left open. [3]

2.3 Packet Sniffer

After the implementation of port filtering now our task is to monitor the rest of the ports. For this purpose, we are going to build a packet sniffer. A packet sniffer can intercept data packets during transmission across a network to capture and then extract useful information from each packet. What are data packets? A data packet is a unit of data transferred over a network. It has some information attached to it. For instance, the IP address, port number, checksum, payload and much more. In our packet tracer we are extracting the Source/Destination IP and Port and the payload. There are two types of packets we are mainly going to work with. TCP packets and UDP packets. We are using the .NET library” Sharp pcap” to capture packets and extract useful information from it. Pcap is made up of an application programming interface (API) for data packet capturing from the network traffic. [4] The most important part is the payload. We are going to compare the payloads against the YARA rules. The rest of the information, related to the IP/port is saved in a log file. This can be used for future forensics. It may give us useful insights about the attacker.

2.4 Yara Rules

YARA (Yet Another Recursive Acronym) is used for malware research and detection. It forms a way for researchers and security experts to describe malware families using textual or binary patterns. As such, rules are then developed that help in the identification and classification of malware by scanning files, processes, or network traffic for matching patterns. There exist three major constituents within a YARA rule: metadata, strings, and conditions. The metadata section contains authors, descriptions, and creation dates for a rule. The strings section enumerates what patterns in the target data to look for; these may be strings or hexadecimal, or regular expressions. The condition section defines what logic controls whether a rule triggers on a target dataset and can contain logical operators or the count of occurrences, or even more complex expressions. Figure 1 shows an example of a YARA rule. This rule comes with a metadata section that provides information on the author, description, and date. Then it has a strings section that lists three patterns: a text string" example malware", a hexadecimal pattern, and a regular expression that matches the word" malware" followed by one or more digits. The condition section at



```
rule ExampleRule {
meta:
 author = "John Doe"
description = "This rule detects Example Malware"
date = "2024-06-28"

strings:
 $text_string = "example malware"
 $hex_string = { E8 ?? ?? ?? ?? 83 C4 04 }
 $regex_string = /malware[0-9]+/
condition:
 $text_string or $hex_string or $regex_string }
```

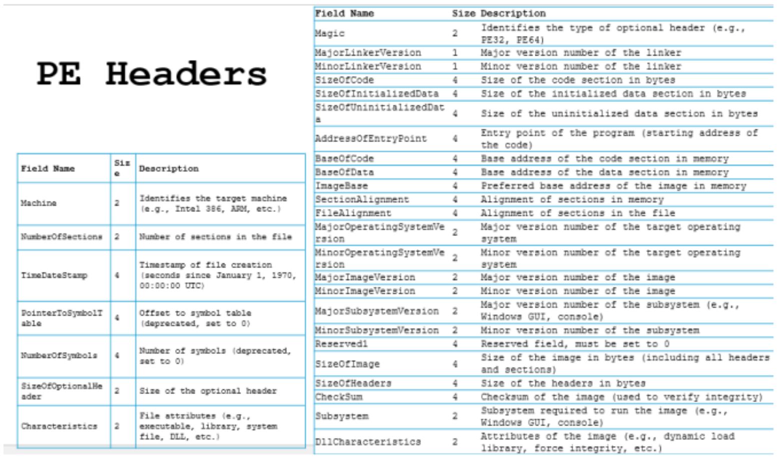
Figure 1 Example of YARA Rules

the end simply states that the rule will match whenever any of the three patterns are found in the target data.

The libyara.net library is a .NET wrapper of the YARA library. It enables a .NET application to access a subset of YARA features. Using this, a .NET developer can include the compilation of YARA rules, scanning, and result processing in his .NET projects and hence use the power of YARA pattern matching in a .NET environment. This is therefore very useful in developing security tools or applications directed toward the detection and analysis of malware or other kinds of patterns in disparate data sources.

2.5 AI/ML

Until now we monitored the packets and the content of the packets. Now is the part when we monitor the files on the system. For example, we keep monitoring our Folder, Downloads, and the moment we detect a PE file we extract the features and check if malicious or not. For this purpose, we are going to use Machine Learning. Machine learning is a branch of Artificial Engineering which is defined as the capability of a machine to imitate intelligent human behavior. There are four types of learning. Supervised learning, Unsupervised learning, Semi-supervised learning, Reinforced learning. Supervised learning is when the present input data is used to reach the result set. Classification and Regression are the types of supervised learning.



PE Headers

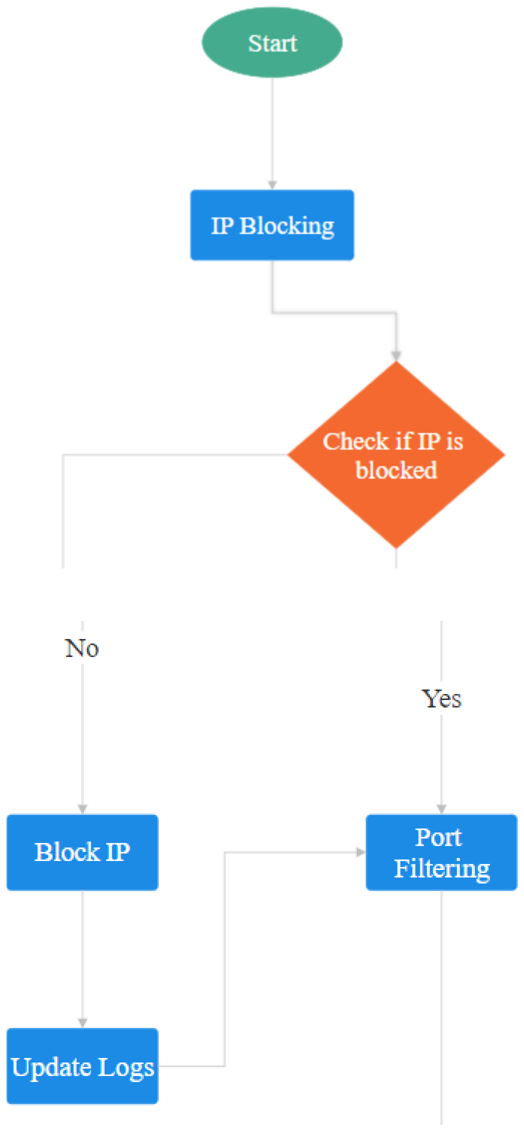
Field Name	Size	Description
Machine	2	Identifies the target machine (e.g., Intel 386, ARM, etc.)
NumberOfSections	2	Number of sections in the file
TimeDateStamp	4	Timestamp of file creation (seconds since January 1, 1970, 00:00:00 UTC)
PointerToSymbolTable	4	Offset to symbol table (deprecated, set to 0)
NumberOfSymbols	4	Number of symbols (deprecated, set to 0)
SizeOfOptionalHeader	2	Size of the optional header
Characteristics	2	File attributes (e.g., executable, library, system file, DLL, etc.)

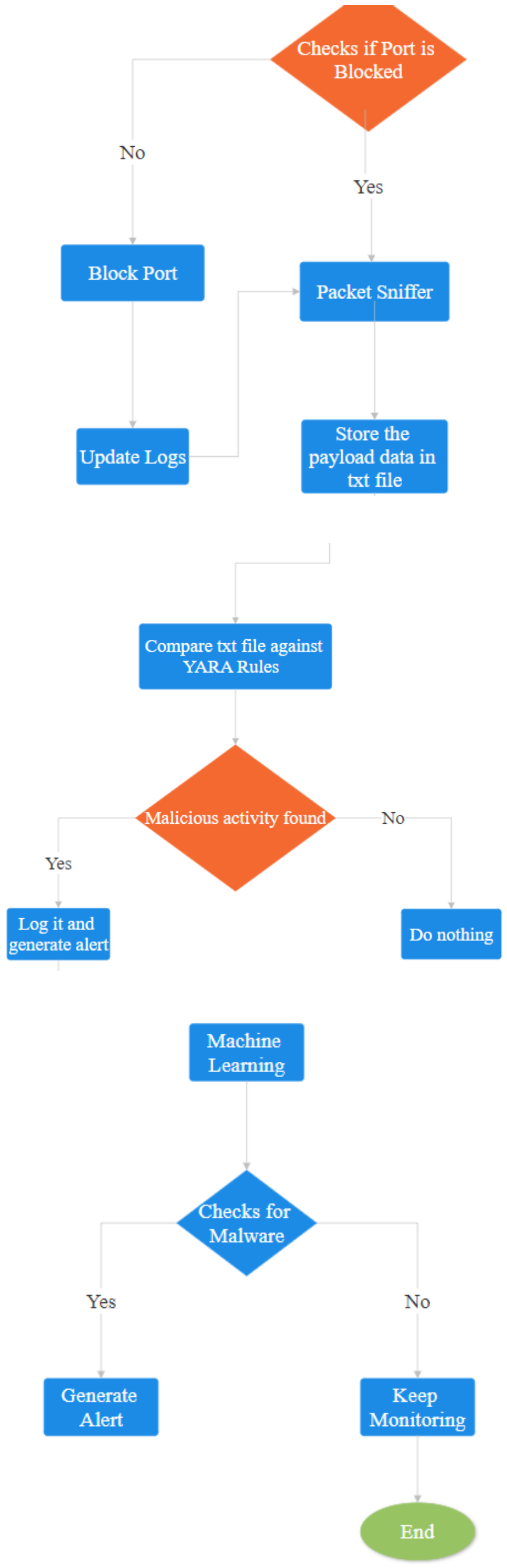
Field Name	Size	Description
Magic	2	Identifies the type of optional header (e.g., PE32, PE64)
MajorLinkerVersion	1	Major version number of the linker
MinorLinkerVersion	1	Minor version number of the linker
SizeOfCode	4	Size of the code section in bytes
SizeOfInitializedData	4	Size of the initialized data section in bytes
SizeOfUninitializedData	4	Size of the uninitialized data section in bytes
AddressOfEntryPoint	4	Entry point of the program (starting address of the code)
BaseOfCode	4	Base address of the code section in memory
BaseOfData	4	Base address of the data section in memory
ImageBase	4	Preferred base address of the image in memory
SectionAlignment	4	Alignment of sections in memory
FileAlignment	4	Alignment of sections in the file
MajorOperatingSystemVersion	2	Major version number of the target operating system
MinorOperatingSystemVersion	2	Minor version number of the target operating system
MajorImageVersion	2	Major version number of the image
MinorImageVersion	2	Minor version number of the image
MajorSubsystemVersion	2	Major version number of the subsystem (e.g., Windows GUI, console)
MinorSubsystemVersion	2	Minor version number of the subsystem
Reserved1	4	Reserved field, must be set to 0
SizeOfImage	4	Size of the image in bytes (including all headers and sections)
SizeOfHeaders	4	Size of the headers in bytes
CheckSum	4	Checksum of the image (used to verify integrity)
Subsystem	2	Subsystem required to run the image (e.g., Windows GUI, console)
DllCharacteristics	2	Attributes of the image (e.g., dynamic load library, force integrity, etc.)

Figure 2 Features of PE Files

Classification is distributing the data into the categories defined on the data set according to their specific features. Regression is the prediction or the conclusion of the other features of the data considering some available features. [3] The portable executable (PE) file format is a windows-based file format for executable files based on the UNIX Common Object File Format. The Windows dynamic linker receives instructions in the PE format on how to load the code contained in the file and the libraries required to run the code in memory. [5] There is a brief description of the features of a PE file in figure 2. Prateek Lalwani, a security specialist, developed a dataset containing characteristics taken from the following: 41,323 Windows binaries (.exe and .dlls), as legitimate files. 96,724 malware files downloaded from the Virus Share website. So, the dataset contains 138,048 lines, in total. We later downloaded several thousand viruses and windows bin files. From the PE files we are extracting 57 features. From the 57 features we selected Twelve (12) important features. Then the Dataset was split into four (4) parts. We selected 3 classifiers, Random Forest Classifier, Gradient Boosting Classifier, and AdaBoost Classifier. The accuracy for Random Forest was 99.4. The model is trained, and the model is then saved and mounted onto our application. We are monitoring the Downloads folder. If there is any EXE file present, then the features are extracted and stested against the trained model. The results are then logged.

3 WORKFLOW


Start
IP Blocking
Check if IP is blocked
No
Yes
Block IP
Port Filtering
Update Logs



Checks if Port is Blocked
No
Yes
Block Port
Packet Sniffer
Update Logs
Store the payload data in txt file
Compare txt file against YARA Rules
Malicious activity found
Yes
No
Log it and generate alert
Do nothing
Machine Learning
Checks for Malware
Yes
No
Generate Alert
Keep Monitoring
End

4 BACKGROUND INFORMATION

HIDS for small networks have come a long way since their establishment. Having been first developed in the late 1980s and early 1990s, HIDS came about as solutions to the increasing requirements that were needed in monitoring and protecting the granular nature of the individual hosts in a network. Such earliest systems were based on analysis against predetermined signatures of known threats through system logs for detecting unauthorized accesses and other malicious activities. Basically, these systems ran on stand-alone computers or servers, offering limited scalability and integration in a network.

In the 2000s, strengthening in computing power and network technologies further fueled the development of HIDS toward being more powerful and adaptive. Heuristics and behavior analysis methods let HIDS detect anomalous patterns and behaviors deviating from normal operations, amplifying its capacity to identify unknown threats. The utility they had in small network environments expanded further by adding integration with centralized management consoles and Security Information and Event Management systems. This way, administrators can monitor and respond to security incidents regarding various hosts centrally.

By the 2010s, HIDS had been incorporated into strategies for complete cybersecurity on small networks. Fueled by the diffusion of both cloud computing and virtualization technologies, the great majority of modern-day HIDS adaptation is for virtual environments to be able to protect across dynamic and distributed infrastructures. Machine learning and artificial intelligence have also been put into today's HIDS, improving their detection abilities against ominous and very fast-evolving threats in real time and reducing false positives.

Today, with innovations in cybersecurity analytics, threat intelligence sharing, and automation, HIDS for small networks has remained as an answer to this diversity of cyber threats, including malware, insider attacks, and focused breaches. It delivers robust protection along with compliance adherence to small businesses without the need for huge infrastructures and resources. In this regard, the further development of HIDS has paid great attention to usability, scalability, and ease of integration with other security tools to ensure that it helps to be effective and usable for the protection of small network environments against increasing cyber threats.

5 RELATED WORK

Research, development, and improvement of HIDS in this realm have been well represented, with several projects addressing effective ascendency and efficient frameworks. The early research in this level was focused on signature-based techniques, which are meant to detect attacks through known patterns describing the behavior of these malicious programs. Programs like Tripwire and the now-retired OSSEC started it off by monitoring system files, and logs are in search for unauthorized access or modification.

Soon after the signature-based approaches' weaknesses became evident—for instance, the inability to detect new or unknown threats—researchers started exploring methods of anomaly-based detection. Amongst such methods, where the baseline of normal system behavior has been created and deviations from this baseline are flagged, several academic and commercial HIDS solutions began their implementation. Research in studies like "A Comparative Analysis of Anomaly Detection Techniques for HIDS" was very instrumental in preparing valuable knowledge on the efficacy of various algorithms and approaches that follow, all giving a view on effectively improving the detection accuracy using machine learning and statistical models.

This is also an area where much of the research has gone into integration with other security mechanisms. Projects dealing with synergy between HIDS and NIDS have already proved to derive considerable advantages from a hybrid approach which joins the detailed monitoring possibilities of a HIDS with the broader and more general network traffic analysis characteristic of NIDS. This hybrid integration can be assessed through systems such as the Snort-HIDS hybrid, which can put together the power of both sorts of IDS technologies into a solution that offers more complete security.

The advent of virtualization and cloud computing brought new challenges and opportunities for HIDS. There are bespoke HIDS under development for virtualized environments, solving the problems of resource efficiency and dynamicity in the case of virtualized infrastructures. Research in this area has focused on the development of lightweight HIDS agents so that that the overall performance is not degraded much by having them deployed within virtual machines, as well as studies to focus on the use of HIDS for attack detection in the cloud environment, including hypervisor vulnerabilities and inter-VM attacks.

6 RESULTS AND DISCUSSIONS

Successfully mastering the techniques mentioned methodology has resulted in a very secure and robust system, replete with multiple barriers against the attacker. Filtering of ports will shut down all the ports that are not necessary, reducing an attack surface immense times and avoiding many forms of attacks over the network. This first step ensures the opening of only very vital channels of communications deemed necessary for the application and avoids exposure to several open ports.

The combination of a packet sniffer with the rules of YARA would provide the necessary muscle in fielding robust network traffic monitoring. We can detect the known malware patterns and suspicious activities easily by analyzing packet payloads against established rules at runtime.

Additionally, the implementation of machine learning to monitor the downloads folder or any other predefined system directories enhances our defense mechanisms. We can train classifiers, especially random forests, in detecting malicious PE files to stop the execution of harmful software before any actual harm is done to the computer. This shall be preventive in nature because, at this stage, whatever malware will be detected would have been neutered way before wreaking havoc.

7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

In summary, we have used the technique of port filtering to filter out all the unnecessary ports. This step will totally mitigate any risk from various open ports. After this first step, we implemented a packet sniffer. From the data extracted from the payload of packets captured via the packet sniffer, we made a comparison against the YARA rules. Then we used Machine Learning to monitor a specific part of our system, in our case the downloads folder. We used different classifiers to determine which random forest was the best. Our model could detect malicious PE files. This will prevent our system from executing malicious PE files from execution. In this paper, we showed the implementation of basic technology and mitigation techniques, Port filtering, Packet sniffing/monitoring, YARA rules, Machine Learning. In future more work can be done on the machine learning algorithms. Deep Learning can be used for better detection of anomalies. HIDS can be combined with a NIDS, this will provide better synergy. Improvements in GUI can be made. Instead of only monitoring PE files, other files like PDF and other documents, database files, zip files etc.

7.2 Future Work

This synergy between HIDS and NIDS has a future recommendation for comprehensive security: this will be the solution that will wed host-based monitoring with network-wide traffic analysis. Counter threats that slash across both hosts and networks will, therefore, at this integration level be viewed for the whole security landscape to aid in better detection and response to complex threats.

This would mean that deep learning algorithms, which will be engineered in the future, will enhance the accuracy and efficiency of anomaly detection. The target currently is to detect even more sophisticated threats which were previously unknown. Extending the monitoring beyond the PE files, such as PDFs, documents, database files, zip files, will secure users against most types of malware or exploits.

Improvement in the graphical user interface will let the system be more user-friendly, open to all, and thus enable administrators to easily manage and monitor this.

8 REFERENCES

[1] M. D. S. Amrit Pal Singh, "Analysis of Host-Based and Network-Based Intrusion Detection System," *Modern Education and Computer Science (MECS),* p. 7, July, 2014.

[2] M. T. M. l. Kuruvilla Mathew, "A Study Of Open Ports As Security Vulnerabilities In Common User Computers," in *International Conference on Computational Science and Technology (ICCST),* Malaysia, 2014.

[3] S. S. A. Özer ÇELİK, "A Research on Machine Learning Methods and Its Applications," *Journal of Educational Technology & Online learning (JETOL),* p. 16, 2018.

[4] A. I. M. Z. M. R. S. Mohammed Abdul Qadeer, "Network Traffic Analysis and Intrusion Detection Using Packet Sniffer," in *Second International Conference on Communication Software and Networks,* India, 2010.

[5] D. S. Collin Connors, "Machine Learning for Detecting Malware in PE Files," 2022.