Honeypots to detect malware and mitigate network traffic attacks using a Game Theory based approach

MSc Internship
Cyber Security

Tanmay Nitin Shinde
Student ID: X18175830

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Tanmay Nitin Shinde
Student ID:	X18175830
Programme:	Cyber-Security
Year:	2019
Module:	MSc Internship
Supervisor:	Michael Pantridge
Submission Due Date:	17/08/2020
Project Title:	Honeypots to detect malware and mitigate network traffic attacks using a Game Theorybased approach
Word Count:	7339
Page Count:	25

Signature:	Tanmay Nitin Shinde
Date:	16th August 2020

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Honeypots to detect malware and mitigate network traffic attacks using a Game Theory based approach

Tanmay Nitin Shinde
X18175830

Abstract

The number of cyber-attacks taking place is increasing day by day in our society. Malware attacks are one such type of attack which infects the system and can cause some unwanted or unpredictable behaviour which may be harmful to its users. DDOS (Denial of Service) attacks are also very common, and can cause a lot of problems. To prevent such attacks and to maintain the integrity of data, some guidelines or steps need to be followed. Implementing a Honeypot is one of such network intrusion detection and prevention technique. There have been numerous different strategies already implemented which identify malware with different ways such as by analysing the system resources used or by simply using YARA rules. In our research we have implemented a honeypot which can log all the connection data received and have also integrated LaikaBoss framework which is a file centric object scanning framework which detects malware by signature detection using static analysis inside our honeypot. We have also implemented a game theory-based technique which can mitigate network attacks such as DOS and DDOS in our honeypot.

1 Introduction

In today's environment, we hear a lot about malware attacks and its effects on an organization hit with a malware attack. Malware is that type of attack which is very dynamic and can reinvent itself as time passes as new vulnerabilities are discovered due to the ever-evolving computer field. There is malware, which is intended to do different tasks, for example different types of malware are ransomware, trojans, worms, etc. All these malwares are designed in a specific way to gain some information about the user or to cause some harm to users. To detect and prevent malware from causing harm there are antivirus programs but if the malware/virus is new and its signatures are not found in the antivirus database then it isn't registered as a virus. Therefore, to help protect organizations and to safeguard their data there is a technique which uses honeypot. Honeypots are basically a dummy copy of a real system or a server which contains valuable information. This is used to attract hackers and to monitor their attack strategies so that the real system could be prevented from such attacks. Honeypots are generally used in big organizations to increase their defence against these hackers as they find newer and newer ways to attack the system.

Honeypots can be used for malware detection by monitoring the system resources used by a program and by keeping logs of the file system changes done by the program to the system. If a honeypot detects that a program can change the file system functions and could perform some malicious activity, it informs the real system so the proper security action could be taken on the real system to improve the system against those attacks. Honeypots could be used to prevent attacks from happening on the real system by warning the real system when an attack takes place on the honeypot. Hence, honeypot works as a type of security measure to safeguard the real system.

Our research uses honeypot basically as an intrusion detection as well as a type of intrusion prevention system. Our implemented honeypot opens up some fake services on specified ports to attract attackers and logs their activity, along with this we have integrated LaikaBoss framework inside our honeypot which is the main part of our honeypot which can detect malware and warns the system automatically whenever a user downloads a malicious file. We have also implemented Zeek IDS inside our honeypot which would add extra layer of preventive measure. To prevent network attacks such as DOS and DDOS we have programmed a game theory script which logs all the activity and uses iptables and implements it whenever an attack is detected.

The paper comprises of different sections, in section 2, critical analysis of the related work, literature review is done, section 3 comprises of the research methodology we approached while doing our research, section 4 comprises of the design specifications of our system, section 5 consists of the implementation of the system, section 6 consists of the evaluation and testing conducted on our system along with the discussion of the testing and section 8 consists of the conclusion and future scope of our research.

2 Related Work

2.1 Honeypots

The growing quantity of cyber-attacks taking place is increasing day by day in our society. To prevent such attacks and to maintain the integrity of data, some guidelines or steps need to be followed. Implementing a Honeypot is one of such prevention techniques. Honeypots are basically a dummy copy of a real system or a server which contains valuable information. This is used to attract hackers and to monitor their attack strategies so that the real system could be prevented from such attacks. Honeypots are generally used in big organizations to increase their defence against these hackers as they find newer and newer ways to attack the system.

There have been numerous researches on honeypots and how they could be helpful in preventing malware as well as network attacks. [1] Liang Huang et al. proposed an approach of DDOS defending strategy using game theory, where they considered the optimal defending strategy by considering the attacker and the defender as two players. The attacking and defence affects are calculated and stored in a GMDCS model. The new defending strategies are formed based on the existing defensive strategies and then the Nash equilibrium is calculated to select the optimal defending strategy. To validate the proposal, they conducted the experiment on a network simulator SSFNet. [2] Whereas Hrishikesh Arun Deshpande proposed the use of virtualized honeypots to prevent distributed denial of service attacks. In their research, the author proposes of having honeypot to mimic a real machine on a single server, have multiple virtual honeypots on a single server. This will reduce the cost as the resources are shared on virtual machines as well as decrease the maintenance cost. He proposes that instead having one honeypot with all the servers, have multiple honeypots with different running servers such as file server, web server, etc. This would be much beneficial than having all the running services on one server but is more resource demanding and harder to setup. In our research we would be using a honeypot on an internal network to log the connection data and integrate it with a malware detection framework to detect malware and any malicious files.

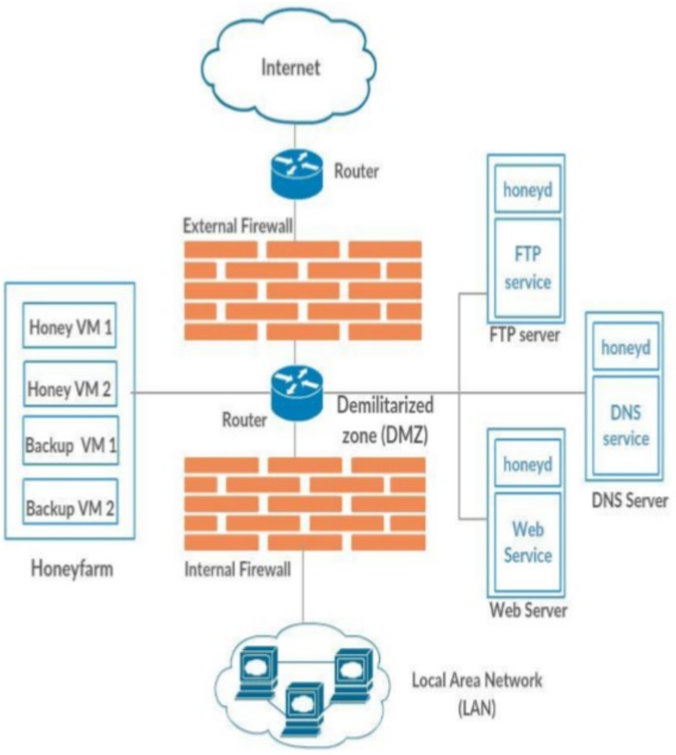


Figure 1: Hrishikesh's HoneyMesh Architecture

Roman Jasek et al. [3] proposed that different types of honeypots such as high-level honeypots, low level honeypots and production honeypots which can be used to detect APTs (Advanced Persistent Threats). These different level honeypots all have their advantages and disadvantages. The research gives a brief description of the process but is hard to implement practically. The basic concept of detection of the APTs is similar to our research. We have also implemented a low interaction honeypot which starts fake services on specified ports to attract attacker.

As per the testing performed by Roman Jasek, over a period, the incidents which are marked in blue are captured by traditional antivirus and anti-malware software, while the ones marked in green are the incidents which are detected by just the honeypots. Also, the incidents marked in red are the incidents which are captured by both the conventional antivirus software s and the honeypots. The research concluded that there is type of malware which could bypass the common solutions i.e. antivirus and can only be captured by honeypots. Such type of malware is the most dangerous.

2.2 Intrusion Detection Systems

Intrusion detection system is a device or an application which monitors a system or a network for malicious activity. If a malicious activity is detected the intrusion detection system informs the administrator of the same. Honeypots are used as intrusion detection

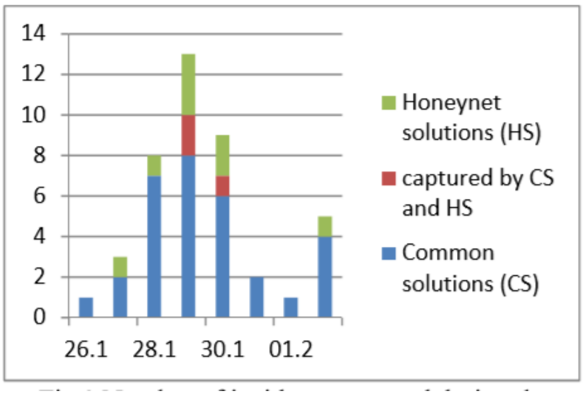


Figure 2: Incidents captured overtime

systems. [4] David Wagner et al. proposed a host-based intrusion detection system which works by static analysis of application behaviour. Static analysis is used to derive a model of application behaviour. This model is used to detect atypical affects such as buffer overflow caused by the attacking application. This research basically focusses on the working of application as they are supposed to be performed with benign intent. The disadvantage of this approach is that it takes a lot of time and analysis to form a base model used for the detection. [5] The research conducted by Simo Kemppainen et al. used a honeypot Kippo to detect malware and attack behaviour of malware in Finland. The research focused on medium level honeypots and its functionality. The gathered research says that the attack consisted of dictionary attack login attempts, attacker location and actions after successful login. The data gathered is helpful for us to use similar strategies to detect and identify malware and its attacks.

The research conducted by [6] Abdullahi et al. proposes a method to properly detect crawler attacks and generally all types of intrusion attacks using honey CAPTCHA which is a framework. The proposed system can be considered as a replacement to the CAPTCHA based IDS which has many issues regarding detection of various forms of intrusion. The proposed research solves those issues. HoneyCaptcha was designed as a means of tricking intruders into thinking it was a web-based application.

2.3 Malware Analysis

Malware analysis is one of the important sections in cyber-security and it has numerous research available for it. Also, there has been an increasing number of major malware and ransomware attacks. Malware is a malicious software which is designed to harm the system in some way. There are different types of malware which affects the performance of the system or injects into the file system. Malware can be detected using analysing the system resources such as the CPU usage, Memory allocation, processes running, etc. Malware can also be detected by analysing the file system for unwanted changes to the files in the system. [7] Yaser Alosefer et al. proposed a method for malware analysis by roaming client honeypots. The roaming honeypots would visit a malicious website or a system and note its effect on the system. A state machine is used to represent the activities done by the malicious web pages on the machine. The states are then passed to a clustering algorithm to summarize the effects of different malicious sites so that solutions could be found to safeguard the real system against such attacks. In our research, we have used [8]

LaikaBoss framework which uses Yara rules and signatures to identify malicious files. We have integrated LaikaBoss in our honeypot such that whenever a user downloads any file through the network, LaikaBoss framework automatically scans the file for malicious signatures. [9] Mirosław Skrzewski et al. have researched and presented the effects of malware from a long-time running honeypot as well as its working and how it spread in the network. The research talks about the importance of selection of proper honeypots, different malware distribution models, malware activity analysis as well as host activity analysis they recorded in a certain amount of time. This research helped us model our honeypot so that it performs the intended tasks and detects malware and keeps its logs safely.

There have been many ransomwares attacks these past years. [10] Chris Moore proposes a method to implement a honeypot to detect ransomware using the honeypot as an intrusion detection system. The research proposes to detect ransomware attack by manipulating Windows Security logs using the services Microsoft File Server Resource Manager feature and EventSentry. While the research done by [7] Dhruvi Vadaviya et al. proposes a simpler technique for malware detection as well as prevention. The research conducted uses a threshold value to detect malicious date on the nodes created on the server. The research combines honeypots, intrusion detection systems and malware analysis in Windows platform. We would not be using the threshold value method in our research rather we would be using a combination of different IDS systems like Zeus integrated with the [8] LaikaBoss framework in our honeypot which would be deployed on the internal network.

There are some types of malware which can spread autonomously. The research conducted by [11] Jianwei Zhuge et al. proposes an integrated toolkit Honeybow which is able to collect autonomous spreading malware in an autonomous manner. Honeybow uses a high interaction honeypot. High interaction honeypots have their advantages and disadvantages. High interaction honeypots have more risk and if exploited the attacker could gain complete access to the system and could perform malicious activities. In our research we would be using a low interaction honeypot which does not allow the attacker to have much interaction with the honeypot as well as the system, but it deceives them into thinking that they are connecting to a real system.

2.4 Game Theory

Game theory is the study of mathematical models which are used for decision making for complex decision-making problems. In cyber-security field it can be used for making proper decisions when a malicious activity occurs in a system. In game theory, the attacker and defender are considered two players and each player makes a move to counter the opponents move. When the counter move is successful, we could say that the game theory had a positive result. Game theory would be used to detect and mitigate network attacks and anomalies such as DDOS, sniffing, etc. [12] Mohammad Hossein Manshaei et al. described in their research the different advantages, disadvantages, and future direction of game theory. Also, the research described future direction of using game theory for cyber security purposes. In our research we would be using game theory to prevent network attacks such as DDOS from happening by using a game theory and an IPtables script which is activated when an attack is detected. [13] J. Markos proposed a system to detect low and high rate DDOS attacks on SDN by analysing the network traffic and by using game theory. Game theory is a modelling technique where the attacker is considered as one player and the defence is considered as another player. Each player acts as per their own strategy. The result of a game theory-based system must be the best possible outcome. In the research conducted by J. Marcos, [13] they propose an approach to mitigate low and high rate DDOS attacks using game theory.

In the research conducted by [14] Sajjan Shiva et al. propose a game theory inspired defence architecture which they called GIDA. They considered the interaction between the attack and defence mechanism as a game played between two players and the actions of one player directly affect the actions of the second player. The research also described a learning algorithm for the game theory to adapt and make it more reliable. In our research we would be taking a simpler approach in game theory and try to apply it to prevent network traffic attacks. The research conducted by [12] Behzad Zare Moayedi et al. proposed a game theoretical approach and use Markov chains to model the system and compute the transition rates between the states based on the skill and preference distributions of hacker classes. The Markov chain then can be used to get the desired safety measures. Using this research, we would get a better understanding of how game theory works and how we should apply it to our own research for maximum result.

2.5 Network Attacks

There have been a lot of network-based attacks on system nowadays. These attacks include but are not limited to Distributed Denial of Service attacks, Botnets, cookie stealing, man in the middle attack and so on. To eliminate some types of these attacks there have been many research proposals. [15] Sherif M. Khattab et al. have proposed a roaming honeypot scheme to mitigate service level distributed denial of service attacks. The honeypots drop connection to an attack machine when it detects unusual traffic and stops acting as a honeypot and acts as an active server. We would be considering a similar approach where we scan the network traffic for some unusual activities and drop traffic if necessary.

The Internet is filled with threats to online security. Many of these threats are basically productive things turned evil. A botnet is an example of a good thing turned evil. Botnets are a bunch of compromised systems in a network. They are controlled by a central bot. These botnets could perform malicious activities like DDOS on a system, sending spam, etc. The basic working of a botnet is that it connects the user's system to the network with many bots which could then perform all types of network malicious activities. [16] Rajab Challoo et al. proposed in their research a solution to detect botnets using Honeypots and P2P botnets. They propose a method to detect the infected honeypot by using a peer to peer structured botnet and using that infected honeypot to detect the original botnets used by the attacker.

DDOS is one of the main attacks conducted frequently. [17] Xupeng Luo et al. in their research they proposed a technique to mitigate DDOS on honeypots. They used a combination of SDN (Software defined networking) and MTD (Moving target defence) architecture to increase uncertainty because of ever-changing attack surface. In the experiment they conducted they were able to prove that SDN based honeypots and MTD can mitigate DDOS attacks. In our research, we would be making the use of IPtables using a game theory-based approach to mitigate DOS and DDOS attacks. [18] Dr. R. Venkatesan et al. conducted a research and proposed a system with virtual honeypot to mitigate DDOS attacks. They also conducted an experiment where they compared and analysed the working of virtual honeypots as compared to physical honeypot systems.

The table below shows the percentage of different types of DDOS attacks. This research is useful for analysing the attacks and creating a proper defence mechanism.

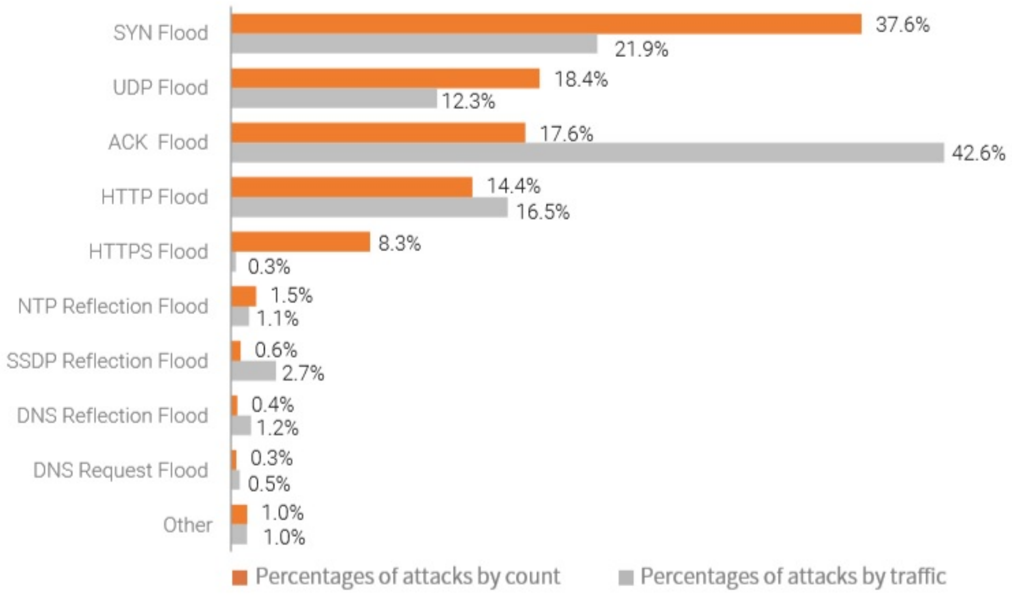


Figure 3: Percentage of DDOS attacks by count and traffic

3 Research Methodology

The research has been conducted after referencing papers, journals, and articles from reputed websites like IEEE, ACM and google scholar.

The research conducted by [13] J. Marcos shows a way to mitigate high level and low-level DOS attacks using game theory. Nash Equilibrium is calculated based on the attack, defence values and the best possible move is calculated. The best move could be whether to block the IP completely, drop packets, or drop the connection completely. We would be using a similar approach and integrate it with a honeypot so that we could add another layer of protection to the real system. We also have another shell script which uses a connection monitoring approach to list the number of connections and if a threshold is reached a particular action is performed to mitigate DDOS attacks. As per the calculations illustrated by J. Marcos, 2017, Below is the equation formed.

$$P_{atk} = w_1^{atk} \cdot E - w_2^{atk} \cdot BC - w_3^{atk} \cdot AC$$

$$P_{def} = -w_1^{def} \cdot E + w_2^{def} \cdot BC + w_3^{def} \cdot AC$$

Figure 4: Game Theory Equation

In the equation above, Patk is the payoff of the attack vector, Pdef is the payoff of the defence vector, Watk and Wdef are the weight of attack and defence respectively. BC is the bandwidth consumed and AC is the attack cost calculated. After calculating the payoff of attack and defence, the best possible move is calculated and performed.

We also have a separate script which contains iptables rules which would be deployed whenever the user wants it to be deployed on the system. [19] This script will not only mitigate attacks as well as prevent any future attacks on the system. These rules are discussed in detail in the following sections. In our research, we have multiple virtual machines which would have different roles such as attacker and the victim. The honeypot will be deployed in our internal network which is inside our firewall.

For Malware detection, we would be using LaikaBoss Framework to scan and detect malware whenever the user downloads any malicious file. [8] The Lockheed Martin Computer Incident Response Team has developed and deployed the LaikaBoss framework for many different organizations. The advantage of LaikaBoss is that it is extremely scalable and can be used for big organizations. We decided to use LaikaBoss framework because there was not much research done on LaikaBoss and being such a powerful tool, it could help organizations protect themselves from different types of attacks. LaikaBoss framework is largely dependant on the YARA rules it has integrated inside of it. YARA rules are the de facto standard signature language to detect and identify malware. They are basically user crafted rules to detect malware based on their functionality or some string. The way Yara works is that it decomposes the file and searches for a particular string or conditions which are specified in the rule and if the condition is satisfied the file is flagged.

YARA is basically a swiss knife to detect and classify malware samples. It uses a set of custom rules which are written by the administrator to detect specific type of malware based on their functionality or author or even its signature. The following is a simple example YARA Rule we created for testing purpose:

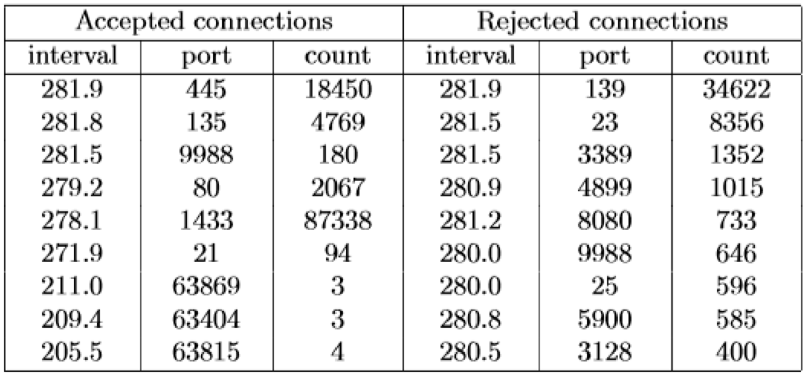
```
rule Example_Test
 {
strings:
$text_string = "Tanmay"
  condition:
       $text_string
 }
```

The above Yara rule is the simplest form of Yara rule where a string is detected. In the example above, a text string is given, and the condition is where the logic of the rule resides. Here the condition is the text string. Hence, when a file containing the string "Tanmay" would be scanned the above rule would be activated and the file would be flagged as malicious.

[20] We would be integrating LaikaBoss along with another framework called Zeek. Zeek is a network analysis tool which sits on the network and logs entire traffic activity and informs the admin if something malicious happens. LaikaBoss is a static analysis tool, to use it for dynamic analysis we have implemented a shell script which scans the file whenever that file is downloaded by the user.

There are different types of honeypots categorized by the level of interaction such as high interaction honeypots, low interaction honeypots and medium interaction honeypots. High interaction honeypots emulate a full operating system or have a real installation of an operating system. An attacker has a lot of freedom on a high interaction honeypot. A low interaction honeypot has limited access for the user. Low interaction honeypots emulate just the network services rather than a full operating system. In our project we have implemented a low interaction honeypot which starts some fake services on ports specified by the admin to deceive the attacker. Medium interaction honeypots are a combination of low interaction and high interaction honeypots. These have some features of high interaction honeypot and low interaction honeypots combined.

The research conducted by Mirosław Skrzewski [9] shows the count of the connections on particular ports on the system overtime. Observing this data, we concluded the most common port to be exploited and connected. Ports 445, 139, 135, 23 and 1433 were among the most connected to. Analysing this data, we can open these ports in our honeypot to attract attacker. The research conducted by Hrishikesh Arun Deshpande,



Accepted connections			Rejected connections		
interval	port	count	interval	port	count
281.9	445	18450	281.9	139	34622
281.8	135	4769	281.5	23	8356
281.5	9988	180	281.5	3389	1352
279.2	80	2067	280.9	4899	1015
278.1	1433	87338	281.2	8080	733
271.9	21	94	280.0	9988	646
211.0	63869	3	280.0	25	596
209.4	63404	3	280.8	5900	585
205.5	63815	4	280.5	3128	400

Figure 5: Connection count on different ports

shows a technique to prevent DDOS attack from happening by using multiple virtualized honeypots called a honeyfarm and routing traffic through this honeyfarm first before reaching the main system. This approach is very time consuming and is also quite extensive as there must be multiple honeypots to be setup which might take a while and also be quite expensive. In their research, Hrishikesh has explained different types of DDOS attacks along with their effect. Rather than having multiple honeypots running at the same time, our system uses a connection monitoring approach to list the number of connections and if a threshold is reached different actions such as blacklisting the IP or dropping the packets or sending RST flags could be performed. The table below shows different types of DDOS attacks.

Table 1: Different types of DDOS attack.

Sr. No.	**Attack**	**Effect**
1.	Smurf Attack	The server is flooded with ICMP packets causing a DDOS attack.
2.	TCP/Syn Flood	Repeated SYN packets are sent to every port on the server thereby flooding the server.
3.	Ping of Death	Malformed or oversized packets are sent on the target server intending to crash, destabilize or freeze the system.
4.	Ping Flood	In this type of attack, the attacker sends multiple ping requests to the server, thereby causing a ddos.
5.	UDP Flood Attack	Forged UDP packets are sent to a port on the target server which responds to destination unreachable ICMP response. When multiple UDP packets are sent, flooding occurs.
6.	Teardrop	In this type of attack, jumbled overlapping packets are sent to the target server thereby, crashing it.
7.	Land Attack	This is a type of attack where packets with identical source and destination are sent to confuse the target server.
8.	Nuke Attack	This sends corrupt and fragmented ICMP packets thereby crashing the target server's operating system.

In our research, we would be using Virtual Machines to test and evaluate our system. We would be using Ubuntu 18.04 LTS as our victim machine which would have our honeypot deployed, and the attacker machine would be Kali which performs DDOS attacks and hosts a sample malware on its Apache server to be downloaded by the victim. The system would be evaluated by performing a DDOS attack on the system by the attacking machine and by downloading a few malware samples to test its detection capability. Below is a use case diagram which shows the functionality of the system and different actors being part of the system.

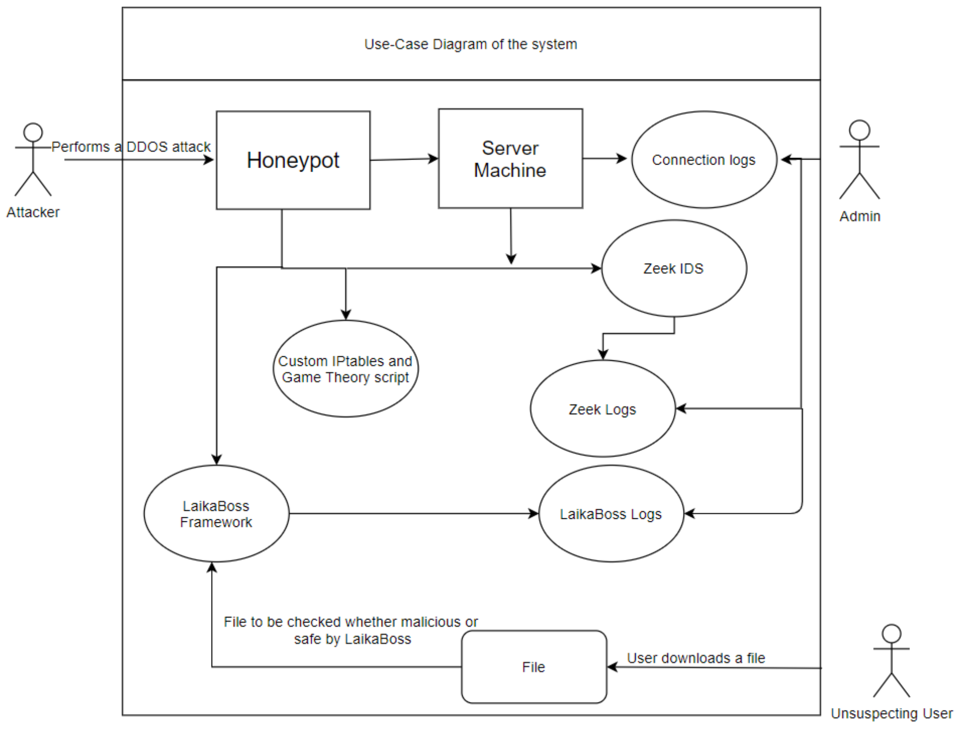


Figure 6: Use Case Diagram of the Proposed System

As we can see in the use case diagram above, there are three actors, the attacker, the admin, and the unsuspecting user. The attacker performs a DDOS attack on the system. Our honeypot uses a custom Iptables and Game theory script to detect and mitigate an ongoing attack and runs another script to prevent such DDOS attacks from happening in the future. Also, our honeypot logs all the connection logs for the admin to analyse later. We have also integrated Zeek IDS into our honeypot which has its own logs which would be monitored by the admin as well. When the unsuspecting user downloads a file that file is scanned by LaikaBoss and checked whether the file is malicious or not. These logs could also be further analysed by the admin.

4 Design Specification

The proposed system consists of two parts with different functions. The first part is the DDOS attack detection, mitigation, and prevention while the second part is the malware detection. The first part is performed when there is an ongoing attack on the system. It uses our game theory script and the IPtables script.

In our first part, game theory script successfully detects an attack and uses Nash Equilibrium to calculate the best possible move to be taken by the system. For DDOS attacks, we also have another shell script which uses a connection monitoring approach to list the number of connections and if a threshold is reached a particular action is performed.

The Game theory works with two layers. Layer 1 detects the attack by monitoring the connection to the system and having a connection threshold. These connection details such as the IP address, destination IP, source and destination ports, protocol type and time are logged and saved into a database server for further analysis. Layer 2 is responsible for analysing the attack and takes a decision using a game theoretical approach. The best possible outcome is calculated, and a decision is made whether to block the ip address or drop the connection. This successfully mitigates an ongoing DOS attack on the system.

[21] We also have a separate IP tables script with a list of fully customizable IPtable rules which are automatically deployed when our honeypot is deployed and an ongoing attack is mitigated. These IPtable commands will prevent from future attacks from happening. Below are some of the important Iptable rules we have used in our script and their brief explanation.

Table 2: IPtable Rules and their Description

`$IPT -A FORWARD -i $WAN -m state --state NEW,INVALID -j DROP`	This does not allow new or unknown network to reach our internal network.

`$IPT -A INPUT -i lo -j ACCEPT`
`$IPT -A INPUT -i $LAN -j ACCEPT`
	This accepts connections from the local machine and local network

`$IPT -N Firewall`
`$IPT -A Firewall -m limit --limit 10/minute -j LOG --log-prefix "Firewall:␣"`
`$IPT -A Firewall -j DROP`
	This is going to handle the packets we don't want to respond to by limiting the amount of logs to 10/min

`$IPT -N Rejectwall`
`$IPT -A Rejectwall -m limit --limit 10/minute -j LOG --log-prefix "Rejectwall:␣"`
`$IPT -A Rejectwall -j REJECT`
	This logs the above packets and informs the sender that the packets were rejected.

`$IPT -A Rejectwall -j REJECT --reject-with icmp-host-unreachable`	This could be used to simulate that the host cannot be reached.

`$IPT -N Badflags`
`$IPT -A Badflags -m limit --limit 10/minute -j LOG --log-prefix "Badflags:␣"`
`$IPT -A Badflags -j DROP`
	This is going to handle the illegitimate packets we do not want to respond to by limiting the amount of logs to 10/min

`$IPT -A INPUT -p tcp --tcp-flags ACK,FIN FIN -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags ACK,PSH PSH -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags ACK,URG URG -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags FIN,RST FIN,RST -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags SYN,FIN SYN,FIN -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags SYN,RST SYN,RST -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags ALL ALL -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags ALL NONE -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags ALL FIN,PSH,URG -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags ALL SYN,FIN,PSH,URG -j Badflags`
`$IPT -A INPUT -p tcp --tcp-flags ALL SYN,RST,ACK,FIN,URG -j Badflags`
	This is a bad flags chain which is going to handle all the TCP bad flags and log and drop them.

`$IPT -A INPUT -p icmp --icmp-type 8 -m limit --limit 1/second -j ACCEPT`
`$IPT -A INPUT -p icmp -j Firewall`
	This is to avoid Ping flood.

`$IPT -A INPUT -i $WAN -p tcp --dport 94 -j ACCEPT`	This is to accept SSH connections
`$IPT -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT`	This accepts related and established connections
`$IPT -A INPUT -j Rejectwall`	Finally, this rule is applied to anything that was not allowed.

LaikaBoss Framework is used in our system to detect malware. LaikaBoss has some configuration files which it uses to identify malicious files and these configuration files can be customized as needed by the admin. Some of these files are disposition files, Yara rules files and the dispatch files. Disposition.yara file consists of all the disposition settings which is located at /etc/laikaboss/modules/dispositioner/. The outcome of a scan is defined by a disposition. If something malicious is found, disposition can be a determining factor and can be used to identify the malware, its priority level, etc.

Yara rules files are the files where the basic Yara rules are stored and are the main part of the LaikaBoss framework. The dispatch.yara file controls the dispatch strings which is located in /etc/laikaboss. It is considered the main part of Laikaboss, or the "brain" of the framework. The dispatch file contains the Yara rules to help define and determine different file types and modules to execute against those file types. Some of the examples of file or object types defined by default, are email, Microsoft Office 2003 and 2007+, PDF, RTF, ZIP, etc.

We chose LaikaBoss Framework for malware detection as LaikaBoss can find malware hidden inside a particular file type. For example, to identify a javascript js file inside a zip, having a dispatch rule as stated below would detect the js file.

```
    rule js_in_zip
{
meta:
flags = "js_in_zip"
condition:
ext_sourceModule contains "EXPLODE_ZIP" and type_is_js
}
```

Such type of malware delivery is the most common technique used by the attackers, and LaikaBoss framework could successfully prevent such attacks from happening. When a zip with js inside it is passed through LaikaBoss with the above dispatch rule, will give the following result.

```
    "flags": [
"dispatch::js_in_zip"
],
...
"source": "CLI",
"filename": "e_zip_47A8A1783B27BAP79A499B.js",
"fileType": ["js"]
```

The scan output shows that there is a new flag hit of "dispatch::js_in_zip" and the filename of e_zip_47A8A1783B27BAP79A499B.js The filename is starting with "e_zip_" because the file originated from the EXPLODE_ZIP module. Also, a flag can be set stating that there was a JavaScript file in a zip as EXPLODE_ZIP was the source module that produced the js file.

LaikaBoss framework can also be used to capture particular strings or to identify malicious emails by writing separate Yara Rules. LaikaBoss is as strong as the Yara rules integrated with it. LaikaBoss can also be used to identify malicious emails. For example, to identify a specific malicious email below rule, need to be implemented in Yara.

```
rule malicious_tester
{
 strings:
 $xmailer_001 = "X-Mailer: Tanmay 1.1"
 $helo_001 = "helo=malicious"
 condition:
 any of ($xmailer_*) and any of ($helo_*)
}
```

The above rule is a simple Yara rule which will check for the X-Mailer and the helo strings. After writing this rule it needs to be updated in the dispatch logic for the type_is_email rule.

```
rule type_is_email
{
 meta:
 scan_modules = "META_EMAIL EXPLODE_EMAIL
SCAN_YARA(rule=/etc/laikaboss/modules/scan-yara/email_rules.yara)"
 file_type = "eml"
 strings:
 $from = "From "
 $received = "\x0aReceived:"
 $return = "\x0aReturn-Path:"
 condition:
 (not ext_sourceModule contains "EXPLODE_EMAIL") and
 (($from at 0) or
 ($received in (0 .. 2048)) or
 ($return in (0 .. 2048)))
}
```

After updating the dispatch logic and creating the new Yara rule, when passing the malicious email, we observe that the email is flagged as malicious with the Yara rule malicious_tester with a flag value.

```
    {
 "scan_result": [
 {
 "order": 0,
 "rootUID": "b345b7-8922-4eqd-1b56-28bceaf32c63431",
 "flags": [
 "yr: malicious_tester "
 ],
 ...
```

Along with LaikaBoss, we have also integrated [22] Zeek IDS into our honeypot and whenever our honeypot is deployed, Zeek server starts monitoring and logging all the data. This data could later be used for analysis by the system admin. Zeek logs are automatically saved /usr/local/zeek/logs.

5 Implementation

The implementation was carried out into three phases. The first phase was to implement a basic honeypot which could start fake services on the specified ports and log the connection details. The second phase was to implement the scripts needed for malware detection. This was done by integrating LaikaBoss using shell scripts to make the whole detection process dynamic. The third phase was to implement a game theory and IPtables script which detects a DDOS attack happening on the system and performs a particular action based on the best possible outcome.

Technologies Used:

Table 3: Technologies Used.

Technology Used	**Version**	**Version**
Python	3.8	Honeypot
MySQL	8.0	Database to store the connection logs
Shell scripts	-	To integrate LaikaBoss, Zeek and Iptables Rules inside our honeypot.

File Structure:

The project consists of 4 python files, 3 shell scripts and a database.

The python files are:

main.py : which starts the honeypot when the user passes the IP address of the system and ports to start the fake services as arguments.

final.py, gt.py, capture.py : these files are used for detecting a DDOS attack and mitigating it using game theory.

script01.sh : This is the program which integrates LaikaBoss and Zeek inside our honeypot and makes the detection process dynamic.

ddos.sh : This is shell script which detects and mitigates DDOS attack by monitoring the connections and having a connection threshold.

Iptables.sh : This is the script with all the customizable IPtable rules as mentioned above which prevents any future attacks from happening on the system.

The implementation has been taken place on a Ubuntu 18.04 LTS virtual machine. All the required dependencies had to be installed prior to the implementation. The detailed implementation and steps to be followed are explained in the Configuration manual.

Below is the figure of different modules of the system and how all these modules are interlinked to the system. Also, the files used by each module are represented in the figure.

The final output of the system is threefold; first when the honeypot is deployed and when an attacker tries connecting to the system, the connection logs are saved and displayed on the honeypot's output screen, second when a DDOS attack is detected, a warning message pops up that the DDOS attack has been detected along with the attacker IP address and the port number and third is when a user downloads a file from the internet, that file is automatically scanned by LaikaBoss and LaikaBoss output is displayed on the honeypots output terminal. Also, when the honeypot is deployed, Zeek server starts monitoring and a message is displayed that the Zeek server has started and is monitoring the activity along with the location of the Zeek logs.

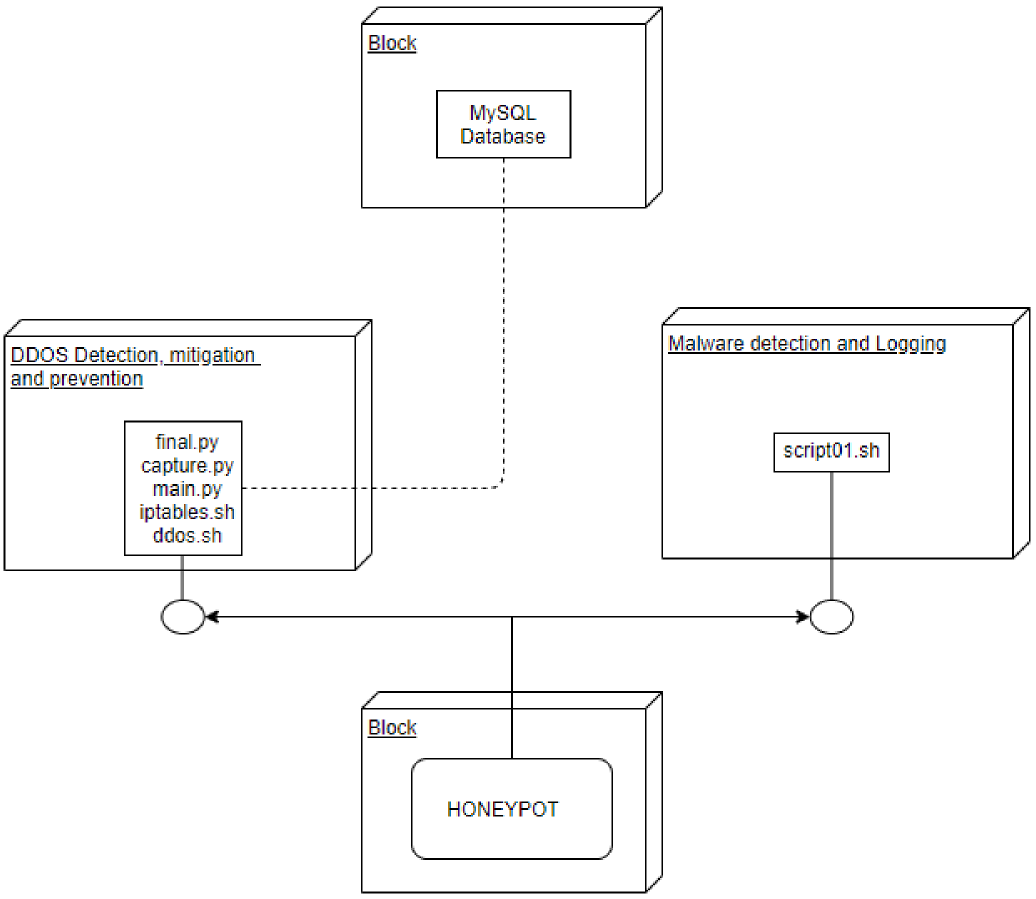


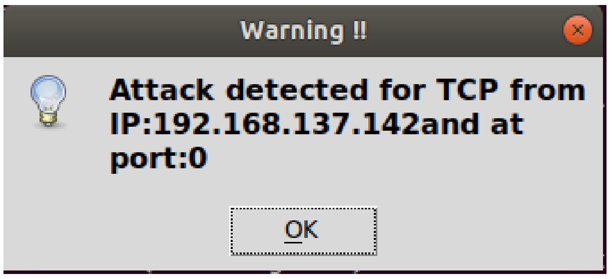
Figure 7: Different Modules of the Proposed System

6 Evaluation

Critical analysis of the functionalities and evaluation was performed on the proposed system. To evaluate the proposed system different Virtual machines were used to conduct different tests. The honeypot was deployed on Ubuntu 18.04 LTS Virtual Machine while the attacks were simulated from two different Kali Virtual machines. All the machines are connected through a virtual network so that they can interact with each other and are completely isolated form the production system so that the production system is not impacted from the experiments conducted. All the virtual machines are running over VMWare 12 on Windows 10 64 bit. DDOS attacks were simulated using the Kali machines to test the detection and mitigation functionality of the proposed system as well as malware samples were hosted to test the malware detection functionality of the system.

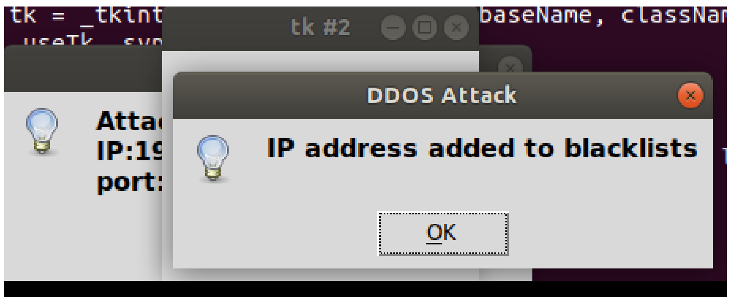
6.1 Detection and Mitigation of DDOS Attacks

DDOS attack has been simulated using a different virtual machine. Kali VM was used to simulate a DDOS attack on our system using hping tool. Hping is a packet generator tool which generates a lot of packets and is mostly used for testing purposes, which can be used for simulating DDOS attack. After initiating the attack, our system detected our attack within 10 seconds and our game theory script performed the best suitable action, in our case it was to block the ip address. The IP address was blocked as it was a high rate attack. After simulating a low rate attack, our system successfully drops the packets and sends the RST flag. Below is the Warning message the system displays when an attack is detected and when the IP address is blacklisted.


Warning !!

Attack detected for TCP from IP:192.168.137.142and at port:0

Figure 8: Warning Message



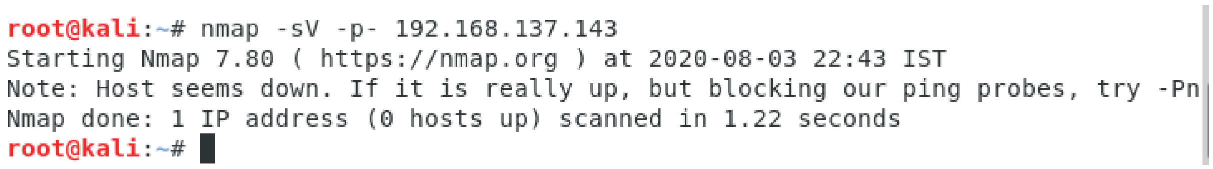
DDOS Attack

IP address added to blacklists

Figure 9: Mitigation

6.2 Prevention of DDOS Attacks

When the honeypot script is deployed and when an attack is detected and mitigated, the IPtables script is also deployed. This IPtables script is used to prevent any such further attacks. To test it, we tried doing the same DDOS attack using hping tool from our attacking Kali VM and every time our system successfully prevented the attack as we got an error as "Host Unreachable". This was because of the IPtables rules we had in our IPtables script which prevents any such further DDOS attacks from happening. To revert back to previous rules the admin has to restore the previous IPtables rules and then the system would be the same as before. Also, the IPtables script is highly customizable and can be customized and changed easily by the admin if need arises and depending on the type of working the organization requires. The below screenshot is when the IPtables script is applied which prevents future DDOS attacks from happening.

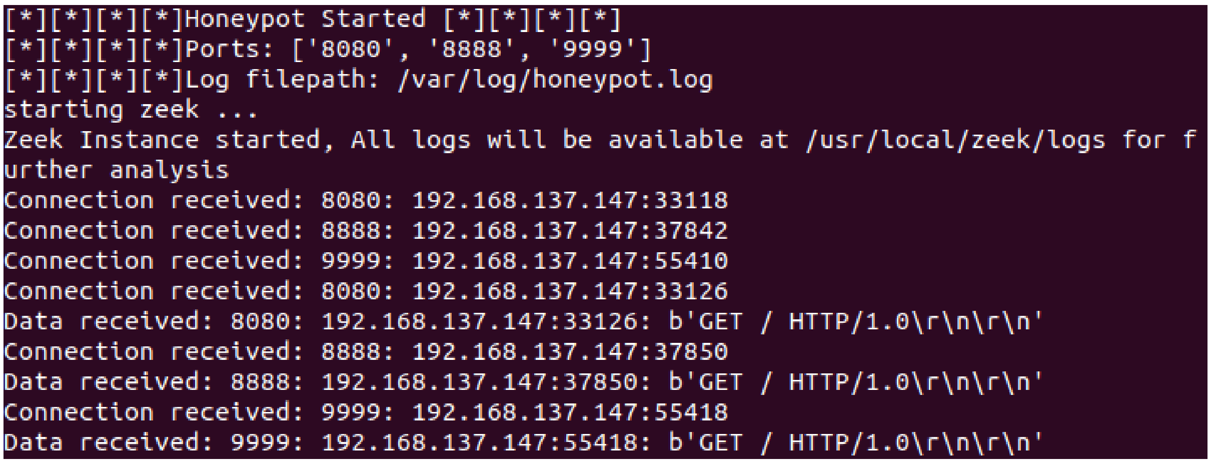


```
root@kali:~# nmap -sV -p- 192.168.137.143
Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-03 22:43 IST
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 1.22 seconds
root@kali:~# 
```

Figure 10: Nmap scan

6.3 Honeypot Logs and Zeek Logs

This was the test performed on our system to check whether our system logs all the network activity and starts the Zeek IDS automatically. As we can see in the image below, Connection details have been displayed on the output of our honeypot. This output was displayed when our attacking machine Kali VM did a reconnaissance scan using nmap to check for open services and ports. After performing the nmap, we observed that the attacking VM can see the fake services open on the ports specified to the honeypot. Also, all the connection logs are successfully logged onto a file as well as displayed on the output screen. The test was conducted several times, while opening different ports and using different machines to connect to the real system, the honeypot worked as intended every time. It logged all the connection activity and the Zeek instance also was started and Zeek started monitoring and logging whenever the honeypot was deployed. The image below shows the output when the honeypot was deployed given the ports 8080, 8888, and 9999. Also, the connection logs are visible as the Kali VM nmap scanned our system. We can also observe that the Zeek instance has been started and Zeek has started monitoring and saving its logs at /usr/local/zeek/logs.

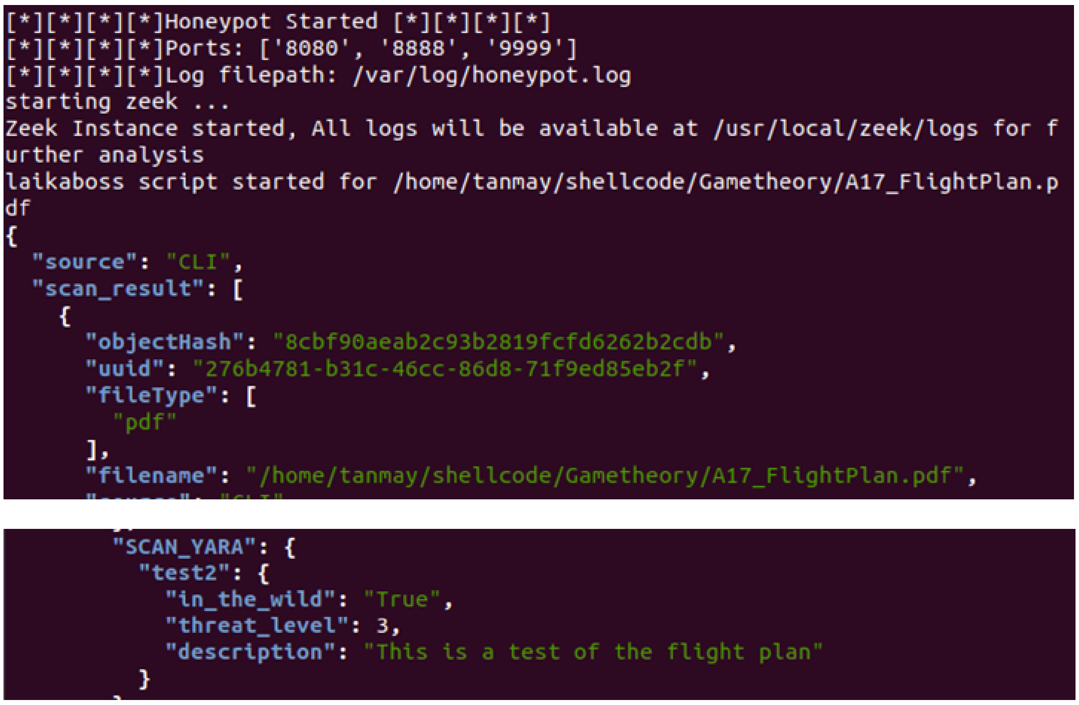


```
[*][*][*][*]Honeypot Started [*][*][*][*]
[*][*][*][*]Ports: ['8080', '8888', '9999']
[*][*][*][*]Log filepath: /var/log/honeypot.log
starting zeek ...
Zeek Instance started, All logs will be available at /usr/local/zeek/logs for f
urther analysis
Connection received: 8080: 192.168.137.147:33118
Connection received: 8888: 192.168.137.147:37842
Connection received: 9999: 192.168.137.147:55410
Connection received: 8080: 192.168.137.147:33126
Data received: 8080: 192.168.137.147:33126: b'GET / HTTP/1.0\r\n\r\n'
Connection received: 8888: 192.168.137.147:37850
Data received: 8888: 192.168.137.147:37850: b'GET / HTTP/1.0\r\n\r\n'
Connection received: 9999: 192.168.137.147:55418
Data received: 9999: 192.168.137.147:55418: b'GET / HTTP/1.0\r\n\r\n'
```

Figure 11: Honeypot Logs

6.4 LaikaBoss Integration and Testing

This was a simple experiment conducted to check whether our LaikaBoss has been successfully integrated in our honeypot. We created a simple Yara rule which detected a simple string found in a pdf file. After deploying our honeypot, we downloaded that file using wget and when the file was downloaded, it was successfully scanned by LaikaBoss and its output was shown in our honeypot's output terminal.



```
[*][*][*][*]Honeypot Started [*][*][*][*]
[*][*][*][*]Ports: ['8080', '8888', '9999']
[*][*][*][*]Log filepath: /var/log/honeypot.log
starting zeek ...
Zeek Instance started, All logs will be available at /usr/local/zeek/logs for f
urther analysis
laikaboss script started for /home/tanmay/shellcode/Gametheory/A17_FlightPlan.p
df
{
  "source": "CLI",
  "scan_result": [
    {
      "objectHash": "8cbf90aeab2c93b2819fcfd6262b2cdb",
      "uuid": "276b4781-b31c-46cc-86d8-71f9ed85eb2f",
      "fileType": [
        "pdf"
      ],
      "filename": "/home/tanmay/shellcode/Gametheory/A17_FlightPlan.pdf",
```

```
        "SCAN_YARA": {
          "test2": {
            "in_the_wild": "True",
            "threat_level": 3,
            "description": "This is a test of the flight plan"
          }
```

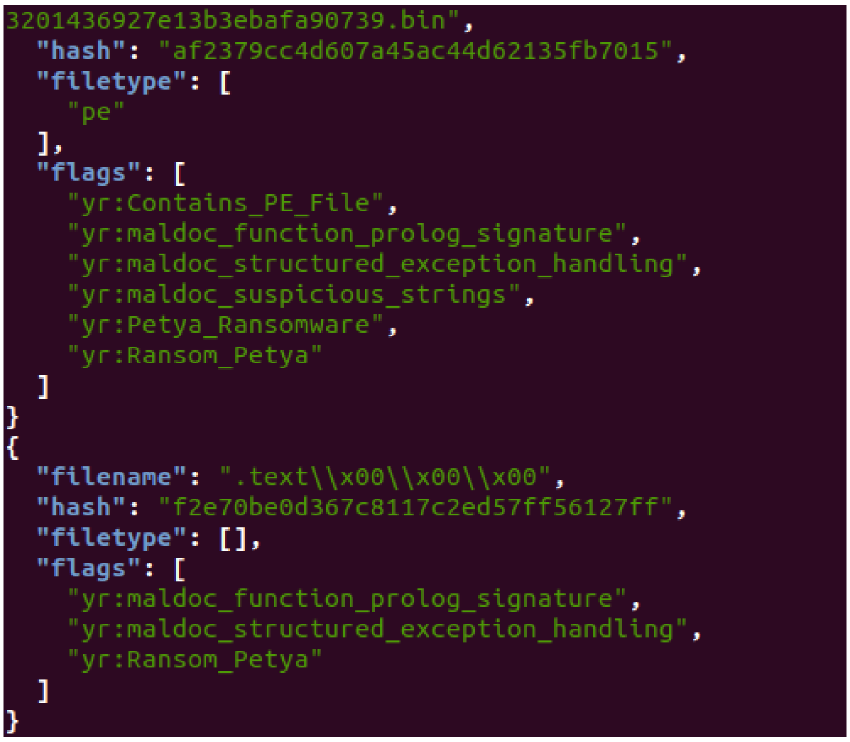
Figure 12: LaikaBoss Integration and Testing

6.5 Detection of a Malicious file

This test was conducted to check whether our system detects a malicious file, in this case it was Petya Ransomware. The attacking VM Kali was used to host the Petya Ransomware file on it's Apache server. When the victim user downloaded that file, our honeypot successfully detected it and displayed its flags.

Several other flags were also displayed which were not specific to the Petya Ransomware. Our system had the Petya Ransomware Yara rule already integrated; this shows us that our systems malware detecting capability is as strong as the Yara rules it has integrated within it.

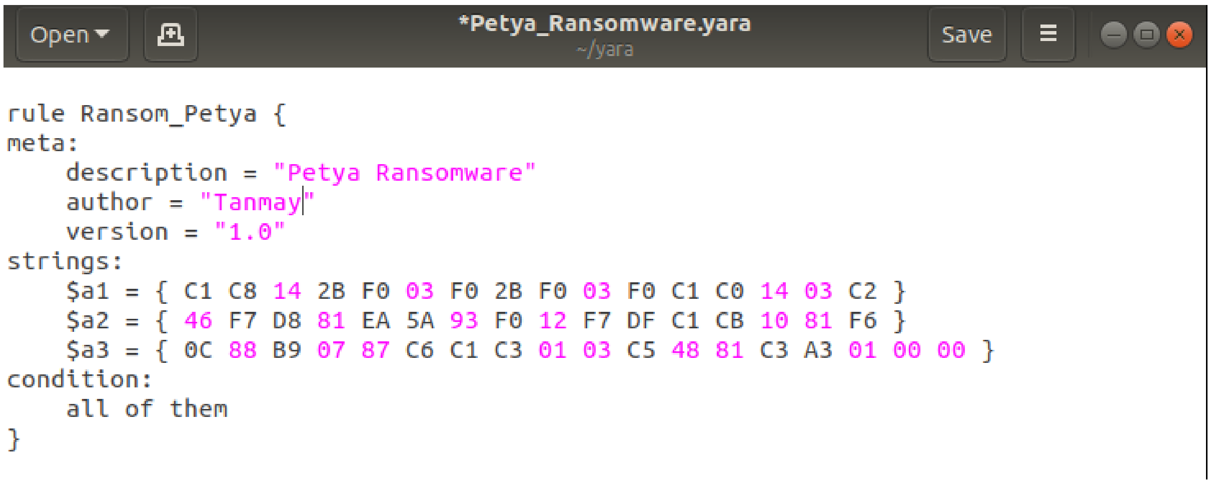
Many tests were performed to detect malicious files and our system successfully detected all of them as long as our system had the corresponding Yara rule. Below is the output of the Petya Ransomware detection done successfully by our honeypot.



```
3201436927e13b3ebafa90739.bin",
  "hash": "af2379cc4d607a45ac44d62135fb7015",
  "filetype": [
    "pe"
  ],
  "flags": [
    "yr:Contains_PE_File",
    "yr:maldoc_function_prolog_signature",
    "yr:maldoc_structured_exception_handling",
    "yr:maldoc_suspicious_strings",
    "yr:Petya_Ransomware",
    "yr:Ransom_Petya"
  ]
}
{
  "filename": ".text\\x00\\x00\\x00",
  "hash": "f2e70be0d367c8117c2ed57ff56127ff",
  "filetype": [],
  "flags": [
    "yr:maldoc_function_prolog_signature",
    "yr:maldoc_structured_exception_handling",
    "yr:Ransom_Petya"
  ]
}
```

Figure 13: Petya_Ransomware Detection

Below is the Yara Rule we wrote after analyzing the Petya Ransomware sample. The strings were present inside the binary file because of which when written in the rule file, the file is detected. This rule detects the Petya Ransomware when it is downloaded by the user and scanned by LaikaBoss.



```
rule Ransom_Petya {
meta:
    description = "Petya Ransomware"
    author = "Tanmay"
    version = "1.0"
strings:
    $a1 = { C1 C8 14 2B F0 03 F0 2B F0 03 F0 C1 C0 14 03 C2 }
    $a2 = { 46 F7 D8 81 EA 5A 93 F0 12 F7 DF C1 CB 10 81 F6 }
    $a3 = { 0C 88 B9 07 87 C6 C1 C3 01 03 C5 48 81 C3 A3 01 00 00 }
condition:
    all of them
}
```

Figure 14: Petya_Ransomware Yara Rule

Table 4: Different Tests Conducted.

Test Cases	Malicious File	Result	Reason
1.	Petya_Ransomware	Detected	Had a corresponding Yara Rule already present in the system
2.	Malicious pdf file in zip	Not Detected	The system did not recognize the zip file and couldn't explode and scan it as there was no dispatch logic regarding the zip present in the system
3.	Malicious pdf file in zip	Detected	After adding the dispatch logic, the system successfully detected the file.
4.	Emotet malware	Not Detected	There was no corresponding Yara Rule present in the system which could detect this type of malware
5.	Emotet malware	Detected	After adding its Yara rule, the system successfully detected the malware.

6.6 Discussion

Multiple experiments were conducted using the proposed system which reflects the different functionalities the proposed system could perform. The first test conducted was the detection and mitigation of DDOS attack, it was concluded from the three experiments conducted that the proposed system successfully mitigated all the DDOS attacks. In the first experiment, as it was a high rate DDOS attack the system blacklisted the IP address. In the second experiment we simulated a low rate DDOS attack and the proposed system mitigated it by sending RST flags. In the third experiment, we again performed a high rate attack using a different attacking VM and the proposed system successfully mitigated it as well. The IPtables had to be restored each time while performing the above experiments as when an attack is detected and implemented the IPtables script is activated and new IPtables are implemented which prevent future DDOS attacks from happening. This was tested by simulating a DDOS attack but the attacking system was not successful to attack the system. This was because of the robust IPtable rules implemented with our system.

Multiple tests were performed to check the malware detection capability of our system and it was found that our system is as strong as the Yara rules and the dispatch logic it has integrated with it. In our experiments, we conducted different test cases, the proposed system successfully detected the malware samples as long as its corresponding Yara rules were in the system. We also had some generic Yara rules which detected malware samples based on the PE file it has hidden inside or which were not completely hidden, but some complex hidden malware needed its corresponding Yara rule. In case 1, Petya Ransomware was tested with our system and our generic Yara rules also detected the file as malicious. We also created a Yara rule to specifically detect Petya and after passing it to our system, that rule was also used along with the other generic rules to detect. In the second case, we had a malicious file inside a zip file and our system could not detect zip file and could not use its different modules. This made our system to not detect the malicious file but once we updated our dispatch logic, our system successfully detected the malicious pdf. Similarly in case 4 and 5, Emotet malware was downloaded and our system could not detect it as it did not have its Emotet's Yara rule , but after adding the Yara rule, our system successfully detected it. With these tests, we can conclude that to keep the system safe and up to date it is necessary to have the Yara rules and dispatch logic updated so that newer malware could also be detected easily. We also had Zeek logs which could be used for the analysis later by the admin. An improvement to the current system would be to have Zeek extracted files pass through LaikaBoss which would add another layer of protection.

7 Conclusion and Future Work

In this paper, an approach to detect malware using LaikaBoss framework on a honeypot was implemented. Along with this, an approach to detect, mitigate and prevent network attacks such as DDOS using a game theoretical based technique was implemented. The implemented system successfully detects malware based on the Yara rules it has integrated within it. It also detects DDOS attacks, logs all the connection activity onto the server and mitigates it by determining a proper action to perform which is either to block the IP or drop the connection. In our experiments, we concluded that the system detects all the malware samples fed into it as long as it has the corresponding Yara rule integrated within it. The system also successfully detected and mitigated the DDOS attacks and prevented any further attacks from happening which is what we concluded from the experiments we conducted. In our case, the honeypot and the system were on the same internal network. This decreases the security of the internal network devices in case the honeypot fails. For future scope, the honeypot could be deployed on the external network i.e. outside the firewall. The LaikaBoss framework which is integrated with the honeypot could receive files from an internal network system using Laikaboss's cloud instance. This could be done to further secure the system and have honeypot running as its own separate dummy system outside the internal network. Also, [23] the Zeek logs and extracted files from Zeek which are saved for further analysis in our system could be fed to the LaikaBoss instance to further increase the security of our system as a whole. For further research, a combination of Zeek, Laikaboss and RITA(Real Intelligence Threat Analysis) could be looked at and integrated in the system.

References

[1] L. Huang and Y. L. Y. Z. Y. L. D. Feng, "A game theory based approach to the generation of optimal ddos defending strategy," 2014.

[2] H. A. Deshpande and N. I. P. Ltd, "Honeymesh: Preventing distributed denial of service attacks using virtualized honeypots," *International Journal of Engineering Research and*, vol. V4, no. 08, p. IJERTV4IS080325, Aug 2015.

[3] R. Jasek, "Apt detection system using honeypots," 2013.

[4] D. Dean and D. Wagner, "Intrusion detection via static analysis," in *2012 IEEE Symposium on Security and Privacy.* Los Alamitos, CA, USA: IEEE Computer Society, may 2001, p. 0156. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SECPRI.2001.924296

[5] S. Kemppainen and T. Kovanen, *Honeypot Utilization for Network Intrusion Detection*, 05 2018, pp. 249–270.

[6] M. Abdullahi, S. Aliyu, and S. Junaidu, "An enhanced intrusion detection system using honeypot and captcha techniques," *FUDMA JOURNAL OF SCIENCES-ISSN: 2616-1370*, vol. 3, no. 3, pp. 202–209, 2019.

[7] D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, and et al., *Clustering Client Honeypot Data to Support Malware Analysis.* Springer Berlin Heidelberg, 2010, vol. 6279, p. 556–565. [Online]. Available: http://link.springer.com/10.1007/978-3-642-15384-6_59

[8] [Online]. Available: https://www.sans.org/reading-room/whitepapers/malicious/paper/38295

[9] M. Skrzewski, *Network Malware Activity – A View from Honeypot Systems.* Springer Berlin Heidelberg, 2012, vol. 291, p. 198–206. [Online]. Available: http://link.springer.com/10.1007/978-3-642-31217-5_22

[10] C. Moore, "Detecting ransomware with honeypot techniques," in *2016 Cybersecurity and Cyberforensics Conference (CCC)*, 2016, pp. 77–81.

[11] J. Zhuge, T. Holz, X. Han, C. Song, and W. Zou, "Collecting autonomous spreading malware using high-interaction honeypots," in *Information and Communications Security*, S. Qing, H. Imai, and G. Wang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[12] B. Moayedi and M. Abdollahi Azgomi, "A game theoretic framework for evaluation of the impacts of hackers diversity on security measures," *Reliability Engineering System Safety - RELIAB ENG SYST SAFETY*, vol. 99, 01 2011.

[13] M. V. O. De Assis, A. H. Hamamoto, T. Abrao, and M. L. Proenca, "A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for dos/ddos mitigation on sdn networks," *IEEE Access*, vol. 5, p. 9485–9496, 2017.

[14] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu, "A survey of game theory as applied to network security," in *2010 43rd Hawaii International Conference on System Sciences.* IEEE, 2010, pp. 1–10.

[15] S. M. Khattab, C. Sangpachatanaruk, D. Mosse, R. Melhem, and T. Znati, "Roaming honeypots for mitigating service-level denial-of-service attacks," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, 2004, pp. 328–337.

[16] R. Challoo, , and R. Kotapalli, in *Detection of Botnets using Honeypots and P2P Botnets.*, vol. 5, 2011.

[17] X. Luo, Q. Yan, M. Wang, and W. Huang, "Using mtd and sdn-based honeypots to defend ddos attacks in iot," in *2019 Computing, Communications and IoT Applications (ComComAp)*, 2019, pp. 392–395.

[18] R. Venkatesan, G. Kumar, and M. Nandhan, "A novel approach to detect ddos attack through virtual honeypot," 07 2018, pp. 1–6.

[19] Jul 2019. [Online]. Available: https://javapipe.com/blog/iptables-ddos-protection/

[20] J. Liburdi, "Laika boss + bro = laikabro (?!)," Feb 2017. [Online]. Available: https://medium.com/@jshlbrd/laika-boss-bro-laikabro-d324d99fddae

[21] B. Musawi, "Mitigating dos/ddos attacks using iptables," *International Journal Of Engineering Technology*, vol. 12, pp. 101–111, 01 2012.

[22] S. Haas, R. Sommer, and M. Fischer, "zeek-osquery: Host-network correlation for advanced monitoring and intrusion detection," *arXiv preprint arXiv:2002.04547*, 2020.

[23] V. Gustavsson, "Machine learning for a network-based intrusion detection system: An application using zeek and the cicids2017 dataset," 2019.