Table of Contents

LISTS OF FIGURES

• CHAPTER

INTRODUCTION

1.1 Introduction

In recent times, there has been a growing interest in security and information protection for network systems. Network systems contain valuable data and resources that must be protected from attackers. Security experts often use honeypots and honeynets to protect network systems. Honeypot is an outstanding technology that security experts use to tap new hacking techniques from attackers and intruders.

According to **Spitzner** (2002),[1] founder of the Honeynet Project, "a honeypot is security resource whose value lies in being probed, attacked, or compromised" (p. 58). It can also be defined as "an information system resource whose value lies in unauthorized or illicit use of that resource" (Spitzner, 2003). In other words, a honeypot is a decoy, put out on a network as bait to lure attackers. Honeypots are typically virtual machines, designed to emulate real machines, act or create the appearance of running full services and applications, with open ports that might be found on a typical system or server on a network (Sahu & Richhariya , 2012).

In this research, a centralized system management called Puppet was used to automate the configuration of four servers. A VMware Virtual Machine was used to implement automated honeypot solutions. The study provided targets with interesting services such as Apache webserver, MYSQL server, File Transfer Protocol (FTP) server and Simple Mail Transfer Protocol (SMTP) server. A centralized Logstash server was used to process and index logs. Elasticsearch was used to store logs. Kibana was used to search and visualize the logs.

1.2 Problem Statement

Attempts by attackers to breach security systems are rising every day. Intruders use tools like SubSeven, Nmap and LoftCrack to scan, identify, probe and penetrate Enterprise systems. Firewalls are put in place to prevent such unauthorized access to the Enterprise Networks. However, Firewalls cannot prevent attacks coming from Intranet.

An Intrusion Detection System (IDS) reviews network traffic and identify exploits and vulnerabilities; it is able to display alert, log event, and e-mail administrators of possible attacks. An Intrusion Prevention System on the other hand makes attempts to prevent known intrusion signatures and some unknown attacks due to the knowledge of attack behaviors in its database. However, an IDS can generate thousands of intrusion alerts every day, some of which are false positives. This makes it difficult for an IDS to detect and identify the actual threats and to protect assets. Thus, human intervention is required to investigate the attacks detected and reported by an IDS (Kaur, Malhotra, & Singh, 2014).

1.3 Nature and Significance of the Problem

Honeypots can dramatically reduce false positives. Honeypots are designed to track illegal activities. This makes it extremely efficient to use honeypots for detecting attacks. Honeypots only collect data from human or processes interactions. Organizations that may log thousands of alerts a day with traditional technologies will only log a hundred alerts with honeypots (Kaur et al., 2014). Honeypots, on the other hand, can easily be used to identify and capture new attacks. New attacks can easily be detected by a honeypot because any illegal activity is an anomaly. Thus honeypots can be used to collect, manage and analyze more attack data.

1.4 Objective of the Study

The objectives of this experiment are:

- to use free and open-source technologies and methods to reduce the amount of manual intervention needed to add to or modify a high- interaction honeypot system suitable for academic research.
- to detect attack patterns on services and come out with solution to mitigate the attacks.

1.5 Research Questions

1. How should open source technologies be used to dynamically add or modify hacking incidences in a high-interaction honeynet system?

2. How should honeypots be made more attractive for hackers to spend more time to provide hacking evidences?

1.6 Definition of Terms

Honeypot: Honey Pot Systems are decoy servers or systems setup to gather information about attackers who intrude into a system.

Tpot: a unique approach to IT automation for discovering, configuring, and managing network infrastructure

Virtual Machines: a virtual machine is a type of computer application used to create a virtual environment, which is referred to as "virtualization." Some types of virtualization let a user run multiple operating systems on one computer at the same time.

HonSSH: a great tool for high-level honeypot interaction.

2. CHAPTER

BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

This chapter discusses the background literature associated with the issues of the design and implementation of effective honeypots and honeynets.

2.2 Background Related to the Problem

Spitzner (2003) [1], introduced two types of honeypots: low-level interaction honeypots are computer software that emulate operating systems and services–usually applied in a production environment within an organization, and high-level interaction honeypots that involve the deployment of actual operating systems on real or virtual machines–it is also used by security research. Also, Spitzner (2003) defined two critical requirements of honeynet architecture: (1) the data control reduced risk by ensuring that once an attacker breaks into honeynet systems, those compromised systems cannot be used to attack or harm other systems, and (2) data capture ensures that security experts can detect and capture all activities performed by the attacker, even if they are encrypted. This study used high interaction honeypots to capture data from real-life systems.

Sobesto et al. (2011) [2] introduced DarkNOC, a management and monitoring tool for complex honeynets. It consists of different types of honeypots as well as other data collection devices. They designed a solution that is able to process large amount of malicious traffic received by a large honeynet and effectively provide a user-friendly web interface to show potential compromised hosts to network security administrators, and also provide the overall network security status. This research used Kibana to implement, a user-friendly web interface, to visually show the attacks eith high incidence. Dittrich (2004) noted that a distributed honeynet can grow in size. A single honeypot cannot effectively monitor and manage attacks. The data analysis of attacks originating from a large number of individual honeypots within a network is difficult. Dittrich recommended the use of Manuka, a database tool with front and back end client applications for resolving this difficult problem. The database includes attributes of systems augment a search for operating system type, version and services installed. He stated that the Manuka tool can only take one person to install honeypots, upload it to a database and rapidly deploy it in a distributed network. Likewise, this study used an open source Puppet automation management tool to deploy servers and services on honeypots without manual intervention.

Weiler (2002) [3] proposed a system that can be applied by large organizations to defend against distributed denial of service attacks. His study relied on honeypot technology that has two advantages. First, the system can defend the operational network of the organization against known distributed denial of service attacks (DDoS) and against new future types of attacks. Second, the system can trap the attacker and record the compromised components to provide evidence for use in a legal action. Weiler said the lessons learned can be implemented in the rest of the network as a protective shield.

Weiler (2002) designed includes a demilitarized zone network that implemented services such as web, mail, ftp and DNS for access by external networks. The local internal network (LAN) of the organization is in another zone protected by a firewall. The infrastructure, then introduced a new system: a honeypot that will mimic the internal network and attract DDoS attackers. Furthermore, Weiler made internal systems in the organization to act as a honeypot. "For example, if the attacker's compromised packets to the webserver of the corporation are detected, the packets go to the honeypot for processing. The reply the attacker gets can be indistinguishable from a real reply of the web server" (Weiler, 2002).

The above design solved three problems, these are: attacks can be detectable, attack packets can be actively directed to the honeypot, and the honeypot is able to simulate the organization's network infrastructure, at least the parts known to the attacker.

Wilson, Maimon, Sobesto, and Cukier (2015) [4] studied the effect of a surveillance banner in the attacked computer system that reduced the probability of commands being typed in the system during longer first system trespassing incidents. The study further stated the probability of commands being typed during subsequent system trespassing incidents, on the same system, is conditioned by the presence of a surveillance banner and by whether commands have been entered by previous trespassing incidents.

Wilson et al.'s (2015) [5] results show that the average number of system trespassing incidents per computer are 4.44 and 4.48 for no surveillance banner and surveillance banner computers, respectively. This is not a statistically significant difference; however, the presence of a banner did affect the seriousness of trespassing incidents. The study further stated that intruders receiving a surveillance banner, that spent at least 50 seconds on the system, are 8% less likely to enter commands into the system than respective intruders that do not receive a surveillance banner. Of those that did not previously enter commands into the system, 38% of those with the surveillance banner and 47% of those with no surveillance banner entered commands during the second trespassing incident. Moreover, of those that do enter commands into the system, 67% of those with the surveillance banner and 63% of those with no surveillance banner entered commands during the second trespassing incident.

The study originates from a randomized controlled trial conducted at a large public university in the United States. Over a 7-month experimental period, 660 target computer systems were deployed after being compromised by system trespassers. These systems produced 2,942 system trespassing incidents (Wilson et al., 2015). Again, they said, each of these systems was randomly assigned to either display a surveillance banner (n=324) or not display a surveillance banner (n=336) upon each entry to the relevant system. The surveillance banner is depicted at left exactly as the intruders saw it, with the message: “This system is under continuous surveillance. All user activity is being monitored and recorded” (Wilson et al., 2015).

2.3 Literature Related to the Problem

Stockman, **Rein**, & **Heile** (2015) used open-source honeynet system to study system banner message effects on hackers. The study was performed for 25 days, gathering data on almost 200,000 events.

According to **Stockman et al.** (2015)[6] , there were 510 logins allowed via the password spoofing, and of the 510 logins, 280 were served a warning banner, and 230 were served no banner at all. The mean duration of attempts for those with the warning banner was 15.29 seconds, and for those without, 23.45 seconds. The median duration of those with the warning banner was 9 seconds, and of those without, 11 seconds. In addition, intruders did nothing while logged in due to the fact that the system recorded only a handful of commands. However, they stated that, the study did not allowed attackers to log in as root, and that might cause lower levels of activity.

Stockman et al.’s (2015) works used the manual intervention in setting up the honeynet. However, this study explored ways to provide targets of greater interest for hackers to attack services such as web or database servers which appear to be resources of actual value for attackers.

Döring (2005) [7] proposed five test cases within which honeypot can be deployed differently in separate environment and explains their individual attributes. According his study, “the amount of attacks occurring in a protected environment are less than the number of attacks coming from the unprotected environment at least they should. Therefore, a comparison of results afterwards needs to focus on the environment.”

The results of Döring’s (2005) research provided a pratical approach for honeypot deployment. He discussed four phases: analysis, development, realization and conclusion phase. According to him, the first phase of development start by gathering knowledge about Honeypots and defining requirements. The development phase concentrates on selecting a particular solution and preparing the requirements for an experiment. The next phase deals with the practical research and empirical analysis and the final phase the gathered results and conclusions are summarized.

Hoque and Bikas (2012) [8] presented an Intrusion Detection System (IDS), using genetic algorithm (GA) to efficiently detect various types of network attacks. Their approached used evolution theory to information evolution in order to filter the traffic data and thus reduce complexity. They used randomly generated population of chromosomes, which represent all possible solution of a problem that are considered candidate solutions. From each chromosome different positions are encoded as bits, characters or numbers.

Similarly, **Jaiganesh**, **Sumathi**, and **Vinitha** (2013) [9] compared two data mining classification algorithms for intrusion detection system, Iterative Dichotomiser2 (ID3) algorithm and C4.5

algorithm. The results presented by Jaiganesh et al. (2013) shows C4.5 algorithm was best suited for intrusion detection, because it uses numeric and nominal data.

In addition, **Stiawan**, **Abdullah**, and **Idris** (2011) [10] presented mapping problem and challenges of Intrusion Prevention System (IPS). They summarized current methods in implementing IPS, the future directions and challenges in reseach field. Stiawan et al. stated in their results that, “Hybrid techniques is one of solution for classification and detection intrusion threat.” The hybrid IPS is able to data capture accuracy and precision for threats.

3. CHAPTER

REQUIRMENT ANALYSIS

3.1 Functional Requirement

The system's configuration for implementing a honeypot necessitates a minimum RAM of 8GB to ensure smooth operation. Additionally, it mandates a substantial 156GB of available disk space for storing captured data. This setup facilitates the honeypot's functionality, enabling efficient capture, analysis, and storage of incoming cyber threats. Adequate resources guarantee seamless operation, ensuring the system effectively identifies attackers' tactics and sources. This configuration is pivotal for bolstering the system's cybersecurity posture, providing valuable insights to fortify defenses against evolving threats while maintaining optimal performance and data handling capabilities.

3.2 Software Requirement

The software requirement involves installing and configuring Tpot, an SSH server, and Pentbox on the system to support the honeypot setup. This includes the latest versions of Tpot, a compatible SSH server for secure access, and Pentbox for comprehensive security testing. Compatibility checks for seamless integration and functionalities between Tpot, SSH server, and Pentbox are essential. The system must enable concurrent operation of these tools, ensuring their compatibility and optimal performance to facilitate efficient honeypot deployment, robust cybersecurity analysis, and proactive threat detection for enhanced system resilience and security.

4. CHAPTER

PROPOSED METHOD

4.1 Configure SSH

Here's a step-by-step breakdown

Steps 1: Giving Kali Machine root permission

Steps 2: Install OpenSSH Server

Steps 3: SSH configuration for a run on persistently

Steps 4: Update-rc.d -f ssh defaults

Steps 5: SSH default keys change

Steps 6: Service SSH start

- **Design SSH Server**

Steps 1: Start SSH Configuration

Steps 2: Include Configurations

Steps 3: Check for additional configurations in /etc/ssh/sshd_config.d/*.conf.

Steps 4: Set SSH Port.

Steps 5: Define the port number for SSH connections.

Steps 6: Define Listening Addresses.

Steps 7: Allow SSH to listen on any address (AddressFamily any) or specific addresses (ListenAddress 0.0.0.0, ListenAddress ::).

Steps 8: Host Keys.

Steps 9: Specify host keys for RSA, ECDSA, and Ed25519 encryption algorithms.

Steps 10: Rekeying.

Steps 11: Set the rekey limit for security purposes.

Steps 12: Logging Settings.

Steps 13: Define the logging facility (SyslogFacility AUTH) and level (LogLevel INFO).

Steps 14: Authentication Settings.

Steps 15: Set the login grace time (LoginGraceTime) and configure root login permissions (PermitRootLogin).

Steps 16: Enable strict modes for security (strictModes).

Steps 17: Specify maximum authentication attempts (MaxAuthTries) and sessions (MaxSessions).

Steps 18: Allow public key authentication (PubkeyAuthentication) and set authorized keys file paths (AuthorizedKeysFile).

Steps 19: Configure Kerberos and GSSAPI options if needed.

Steps 20: Enable or disable password authentication (PasswordAuthentication and KbdInteractiveAuthentication).

Steps 21: Use PAM.

Steps 22: Utilize PAM for authentication and session processing (UsePAM).

Steps 23: X11 Forwarding.

Steps 24: Allow or disallow X11 forwarding (X11Forwarding).

Steps 25: Print Message of the Day.

Steps 26: Decide whether to print the MOTD (PrintMotd).

Steps 27: Accept Environment Variables.

Steps 28: Accept specific environment variables (AcceptEnv LANG LC_*).

Steps 29: Define SFTP Subsystem.

Steps 30: Specify the SFTP subsystem (Subsystem sftp /usr/lib/openssh/sftp-server).

Steps 31: User-Specific Settings.

Steps 32: Optionally, define settings for specific users using the Match directive.

4.2 Configure Pentbox

Steps 1: Clone from GitHub

Steps 2: Connect with SSH Server

Steps 3: Design ssh_config

Connect SSH Server through the following steps:

Steps 1: Start SSH Client Configuration

Steps 2: Include Configurations

- Check for additional configurations in /etc/ssh/ssh_config.d/*.conf.

Steps 3: Host Configuration

- Apply configurations to all hosts (Host *).

Steps 4: Set SSH Client Options

- **ForwardAgent:** Decide whether to forward the authentication agent connection.
- **ForwardX11:** Enable or disable X11 forwarding.
- **ForwardX11Trusted:** Specify if X11 connections should be trusted.
- **PasswordAuthentication:** Allow or disallow password authentication.
- **HostbasedAuthentication:** Enable or disable host-based authentication.
- **GSSAPIAuthentication:** Enable or disable GSSAPI authentication.
- **GSSAPIDelegateCredentials:** Decide whether to delegate GSSAPI credentials.
- **GSSAPIKeyExchange:** Allow or disallow GSSAPI key exchange.
- **GSSAPITrustDNS:** Specify if DNS is trusted for GSSAPI.
- **BatchMode:** Enable or disable batch mode.
- **CheckHostIP:** Decide whether to check the host IP.
- **AddressFamily:** Specify the address family.
- **ConnectTimeout:** Set connection timeout.
- **StrictHostKeyChecking:** Define strictness of host key checking.
- **IdentityFile:** Set identity files for public key authentication.
- **Port:** Define the SSH port.
- **Ciphers:** Specify allowed ciphers for encryption.
- **MACs:** Set preferred message authentication codes.
- **EscapeChar:** Define the escape character.
- **Tunnel:** Enable or disable tunneling.

- **Tunnel Device:** Specify the tunnel device.
- **PermitLocalCommand:** Allow or disallow local commands.
- **VisualHostKey:** Enable or disable visual host key display.
- **ProxyCommand:** Define a proxy command.
- **RekeyLimit:** Set rekey limit.
- **UserknownHostsFile:** Define user-known hosts file.

Steps 5: Set Environment Variables

- **SendEnv:** Send specific environment variables.

Steps 6: Hash Known Hosts

- Decide whether to hash known hosts.

Deploy Pentbox by the following commands:

Steps 1: cd pentbox

Steps 2: cd pentbox-1.8

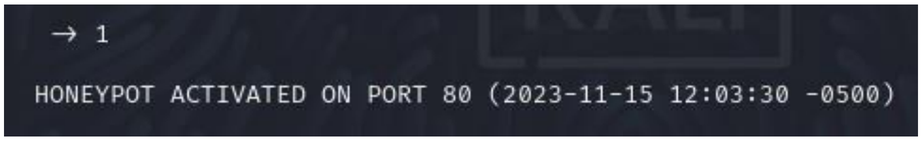
Steps 3: ls

Steps 4: ruby pentbox.rd

Steps 5: Press-2 for 2-Network tools

Steps 6: Press-3 for 3-Honeypot

Steps 7: Select-1 for 1-Fast Auto Configuration



```
 -> 1

HONEYPOT ACTIVATED ON PORT 80 (2023-11-15 12:03:30 -0500)
```

Fig.1: Honeypot Activate

Honeypot Activate

This indicates activating a honeypot on port 80, potentially mimicking a web server to attract and analyze incoming network traffic.

4.3 Configure Tpot

Install and connect with SSH Server

Install Tpot and modify its code to align with our SSH server settings. Adjust configurations for compatibility, enabling Tpot to function with the SSH server seamlessly. This setup ensures the generation of IP and web data, transforming it into an operational system for comprehensive security analysis.

To link Tpot with the system's SSH server, configure Tpot's settings to utilize the SSH server for remote access. Ensure SSH server compatibility, set corresponding credentials, and adjust Tpot's configurations to communicate and operate seamlessly with the SSH server, enabling secure remote access and monitoring.

Here's a step-by-step breakdown:

Step 1: Initial Setup and Checks

- Initialization: Set up initial variables and constants.
- Check Installer: Verify if the installer has already been executed before.
- Define Functions: Define various functions that will be used in the script.

Step 2: Function Definitions

- **fuBANNER:** Create banners for display.
- **fuRANDOMWORD:** Generate random words for hostnames.
- **fuGOT_ROOT:** Check if the script is running as root.
- **fuCHECKPACKAGES:** Check and install necessary packages.
- **fuCHECKNET:** Check the availability of remote sites.
- **fuGET_DEPS:** Install T-Pot dependencies.
- **fuCHECK_PORTS**: Check for active services.

Step 3: Pre-Installer Phase

- **Check Root:** Ensure the script is run as root.
- **Check Packages:** Verify and install required dependencies.
- **Validate Debian Release:** Ensure the supported Debian release version.
- **Parse Command Line Arguments:** Parse and validate command line arguments.

Step 4: Prepare Installer Environment

- **Read Configuration File:** Load configuration if provided.
- **Prepare Environment:** Set up the installation environment.
- **Proxy Configuration:** Set up proxy settings if required.
- **Installer User Interaction:** Interact with the user for installation preferences.

Step 5: Installation Phase

- **Dependency Installation:** Install necessary dependencies.
- **Web User Credentials Setup:** Create credentials for web access.
- **SSL Certificate Generation:** Generate SSL self-signed certificates.
- **NTP Server Setup:** Configure NTP server settings.
- **802.1x Networking Configuration:** Set up 802.1x networking.
- **SSH Roaming Deactivation:** Disable SSH roaming for security.
- **Example Configuration:** Provide an example wireless configuration.

Step 6: Error Logging

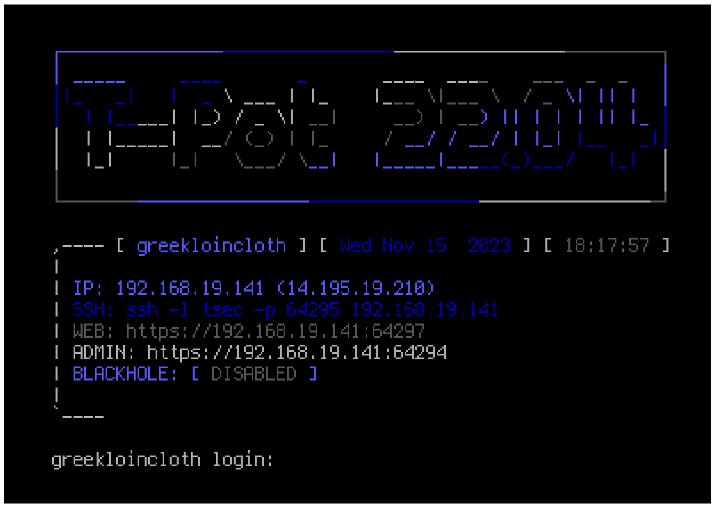
- Redirect Error and Log Output: Direct errors and log script output.

Step 7: Execution

- Execute Script: Run the script to perform the defined steps.

4.4 Generating IP & WEB

Develop user credentials, IP, and web login for website access. Employ robust security measures to analyze and thwart potential attacks, enhancing the overall protection.



```
,---- [ greekloincloth ] [ Wed Nov 15  2023 ] [ 18:17:57 ]
|
| IP: 192.168.19.141 (14.195.19.210)
| SSH: ssh -l tsec -p 64295 192.168.19.141
| WEB: https://192.168.19.141:64297
| ADMIN: https://192.168.19.141:64294
| BLACKHOLE: [ DISABLED ]
|
`----

greekloincloth login:
```

Fig.2: Generate Tpot for IP and WEB

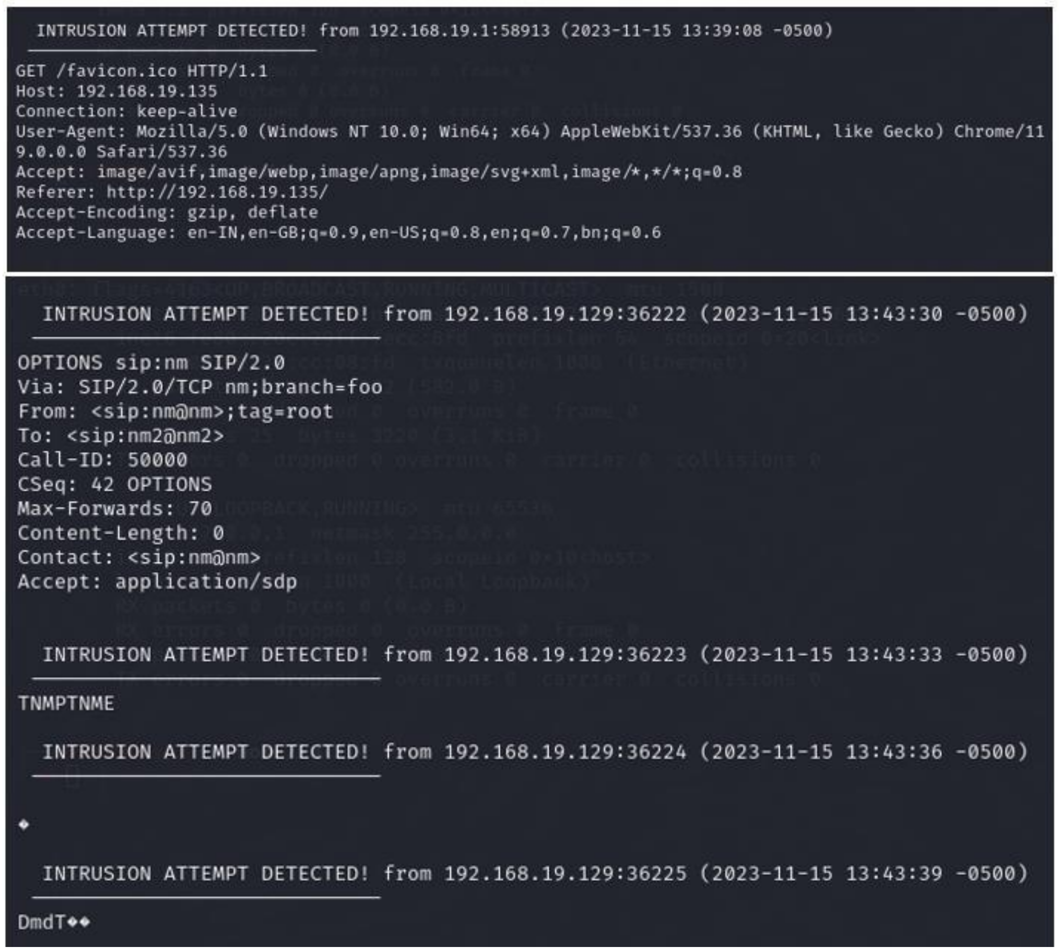
Generate Tpot for IP and WEB

This indicates that the Pentbox tool generates reports for IP and web analysis, revealing IP addresses and web servers for detailed examination of results.

DATA ANALYSIS AND RESULTS

Pentbox Result:

Honeypots proactively identify attackers by capturing their IP addresses, sources, and ports, allowing for comprehensive threat analysis. This information empowers proactive security measures, preventing potential attacks and bolstering the overall resilience of the system, ensuring a robust defense against evolving cybersecurity threats.



```
  INTRUSION ATTEMPT DETECTED! from 192.168.19.1:58913 (2023-11-15 13:39:08 -0500)
 -----------------------------
GET /favicon.ico HTTP/1.1
Host: 192.168.19.135
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/11
9.0.0.0 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8
Referer: http://192.168.19.135/
Accept-Encoding: gzip, deflate
Accept-Language: en-IN,en-GB;q=0.9,en-US;q=0.8,en;q=0.7,bn;q=0.6
```

```
  INTRUSION ATTEMPT DETECTED! from 192.168.19.129:36222 (2023-11-15 13:43:30 -0500)
 -----------------------------
OPTIONS sip:nm SIP/2.0
Via: SIP/2.0/TCP nm;branch=foo
From: <sip:nm@nm>;tag=root
To: <sip:nm2@nm2>
Call-ID: 50000
CSeq: 42 OPTIONS
Max-Forwards: 70
Content-Length: 0
Contact: <sip:nm@nm>
Accept: application/sdp

  INTRUSION ATTEMPT DETECTED! from 192.168.19.129:36223 (2023-11-15 13:43:33 -0500)
 -----------------------------
TNMPTNME

  INTRUSION ATTEMPT DETECTED! from 192.168.19.129:36224 (2023-11-15 13:43:36 -0500)
 -----------------------------

◆

  INTRUSION ATTEMPT DETECTED! from 192.168.19.129:36225 (2023-11-15 13:43:39 -0500)
 -----------------------------
DmdT◆◆
```

Fig.3: Pentbox Result

Pentbox Results

This indicates that the Pentbox identifies and prevents attacks by offering insights into potential security threats, enabling proactive measures to thwart malicious activities effectively.

Tpot:

The experimental investigation on the honeypots was performed for 10 days.During this time some attacks captured by the honeypot The data analysis investigated the source IP addresses, attacks, countries, daily attacks, passwords, usernames and source ports.

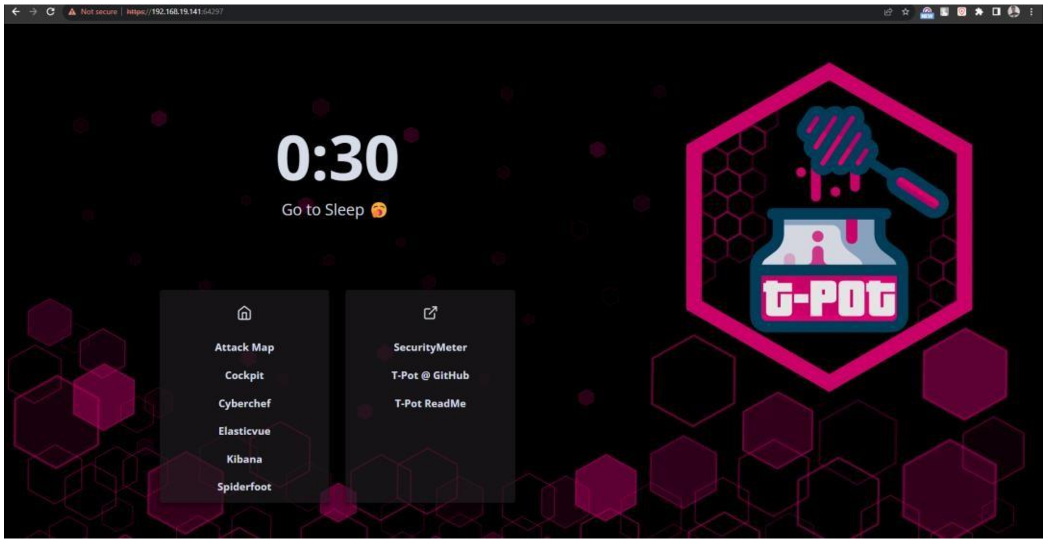
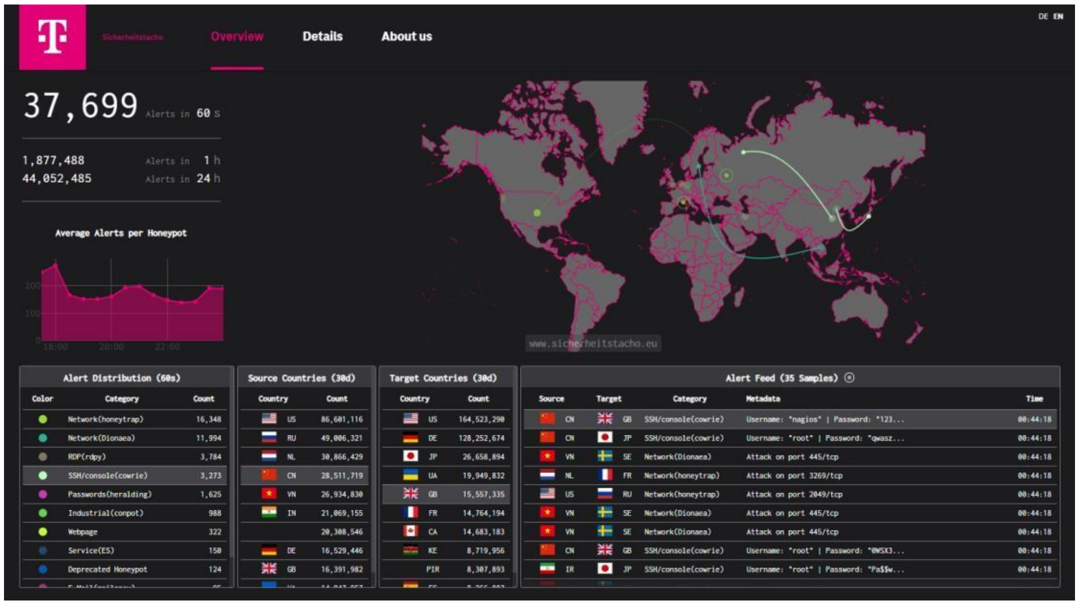


Fig.4: Tpot Main Page

Tpot Main Page

This indicates that The TPOT main page is a centralized hub offering intuitive navigation and access to essential tools and features for effective cybersecurity analysis.



```
37,699   Alerts in 60 s
1,877,488    Alerts in 1 h
44,052,485   Alerts in 24 h
```

Alert Distribution (60s)

Color	Category	Count
●	Network(honeytrap)	16,348
●	Network(Dionaea)	11,994
●	RDP(rdpy)	3,784
●	SSH/console(cowrie)	3,273
●	Passwords(heralding)	1,625
●	Industrial(conpot)	988
●	Webpage	322
●	Service(ES)	150
●	Deprecated Honeypot	124

Source Countries (30d)

Country	Count
US	86,601,116
RU	49,006,321
NL	30,866,429
CN	28,511,719
VN	26,934,830
IN	21,069,155
	20,308,546
DE	16,529,446
GB	16,391,982

Target Countries (30d)

Country	Count
US	164,523,290
DE	128,252,674
JP	26,658,894
UA	19,949,832
GB	15,557,335
FR	14,764,194
CA	14,683,183
KE	8,719,956
PIR	8,307,893

Alert Feed (35 Samples)

Source	Target	Category	Metadata	Time
CN	GB	SSH/console(cowrie)	Username: "nagios" | Password: "123...	00:44:18
CN	JP	SSH/console(cowrie)	Username: "root" | Password: "qwasz...	00:44:18
VN	SE	Network(Dionaea)	Attack on port 445/tcp	00:44:18
NL	FR	Network(honeytrap)	Attack on port 3269/tcp	00:44:18
US	RU	Network(honeytrap)	Attack on port 2049/tcp	00:44:18
VN	SE	Network(Dionaea)	Attack on port 445/tcp	00:44:18
VN	SE	Network(Dionaea)	Attack on port 445/tcp	00:44:18
CN	GB	SSH/console(cowrie)	Username: "root" | Password: "@WSX3...	00:44:18
IR	JP	SSH/console(cowrie)	Username: "root" | Password: "Pa$$w...	00:44:18

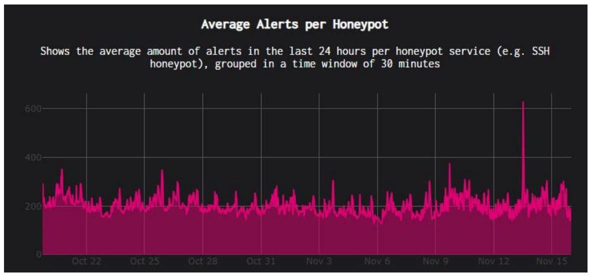
Fig.5: Analyze Worldwide Attack

Analyze Worldwide Attack:

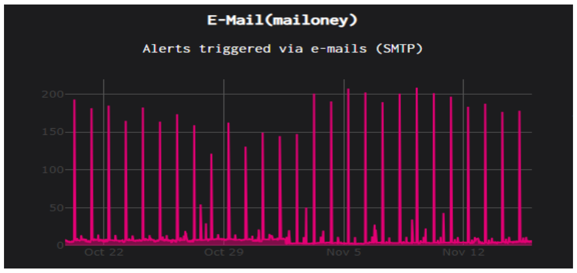
This indicates that T-Pot detects and maps worldwide cyber attacks, identifying countries and their sources using IPs. The visualizes and analyzes this data to enhance cybersecurity insights and threat mitigation strategies.

Honeypot captures global alerts, providing a comprehensive overview of cybersecurity threats worldwide. This data aids in understanding and fortifying defenses against emerging risks for a more secure digital environment.

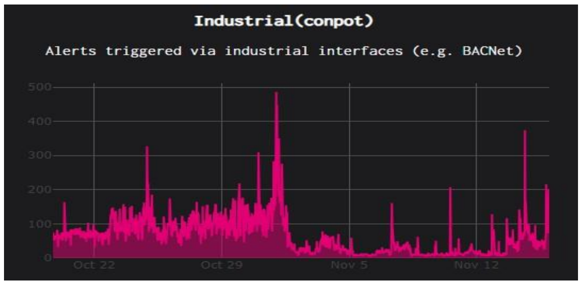
Figure 6 : Report Analysis



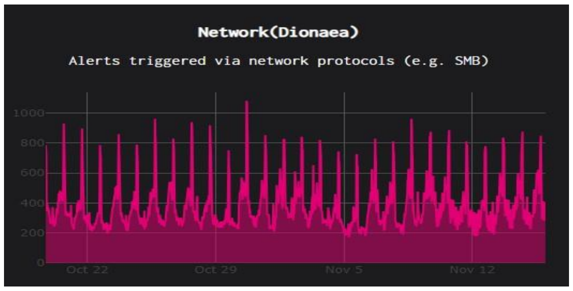
(a)



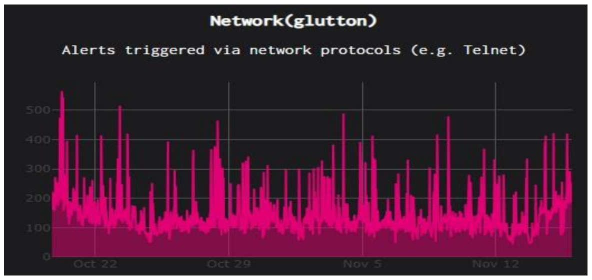
(b)



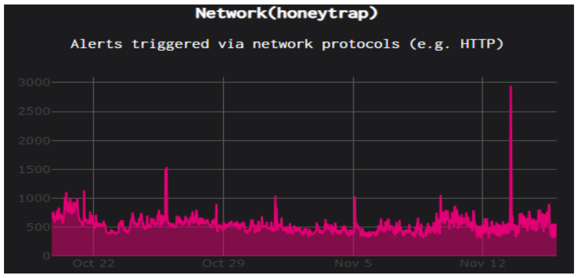
(c)



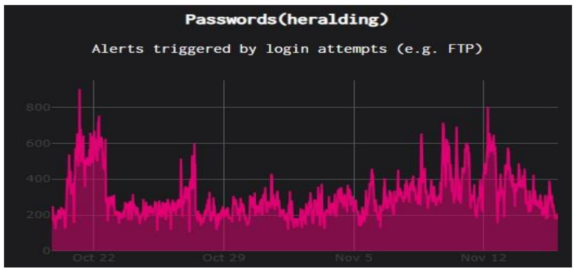
(d)

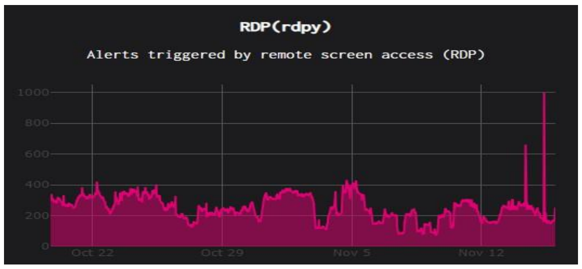


(e)



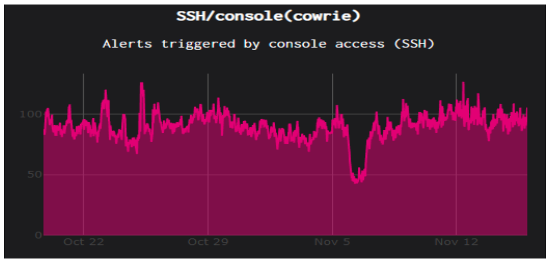
(f)

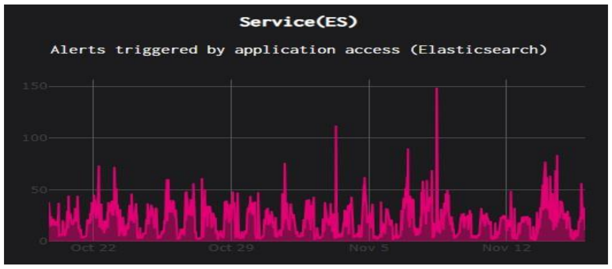




(g)

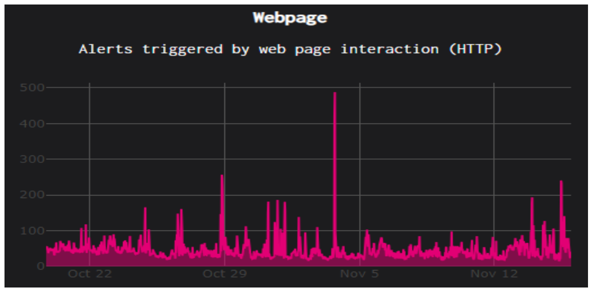
(h)





(i)

(j)



(k)

In the analysis report, multiple figures delineate different alert categories. Figure 6:(a) illustrates the average alerts identified on specific ports, offering insights into their frequency or patterns. Figure 6:(b) corresponds to email alerts, likely indicating potential security threats or suspicious activities related to email communications. Following this, Figure 6:(c) showcases alerts attributed to Conpot, a honeypot framework, while Figure 6:(d) details Dionaea alerts, highlighting instances of malware or intrusions targeting this honeypot. Network through 16 continue the trend, delineating alerts from Figure 6: (e) Network Glutton, Figure 6:(f) Network Honeytrap, Figure 6:(g) Passwords/Heralding, Figure 6:(h) RDP/rdpy, Figure 6:(i) SSH/console cowrie, Figure 6:(j) Service ES, and Figure 6:(k) Webpage, providing a comprehensive breakdown of distinct alert sources or types identified in the analysis.

CONCLUSION & FUTURE WORK

Conclusions:

This chapter summarizes the methodology of the experiments used in the thesis. The implications of the research results from the use of a real-time honeynet system to detect and prevent attacks are discussed.

The research designed and implemented a real-time Honeynet system for detecting and preventing system attacks. System services on SSH Server, MYSQL, FTP and SMTP were used to lure attackers.

The experiment was conducted for 30 days. During this period, some attacks were captured. The results of this study shows the IP addresses from the world, the countries of origin of the attacks, attacks by day, passwords and usernames used by the intruders, and the sources port used for the attacks.

Future Work:

There is more research work that can be conducted to reduce the amount of manual intervention required to add to or modify new honeypots to the honeynet system. In particular, further study can pursue the use of a centralized system management called Ansible to capture and record all activities performed on the honeypots. The design of countermeasure algorithms for redirecting all real attacks to an isolated honeypot for visual inspection by network administrator is an intesting future areas for research. Further studies are required to establish threshold boundary values of each type of network attacks.

APPENDIX

1. Technical Specifications: Detailed hardware and software requirements for the real-time honeypot system.

2. System Architecture Overview: Visual diagrams illustrating the system's architecture, components, and network interactions.

3. Simulation Scenarios: Examples demonstrating the system's functionality in detecting and responding to various cyber attack scenarios.

4. Data Collection and Analysis Methods: Brief outline of methodologies used for data collection, analysis, and threat intelligence extraction.

5. Performance Metrics: Key metrics established to measure the system's performance and effectiveness in threat detection.

6. User Interface Design: Mock-ups or descriptions showcasing the user interface for cybersecurity professionals.

7. Compliance and Ethical Considerations: Summary addressing legal compliance, ethical considerations, and privacy safeguards.

8. Budget Overview: High-level breakdown of projected costs and resource allocation for system development and maintenance.

9. References: List of all sources and references used throughout the project for further reading and validation.

This condensed appendix provides a snapshot of essential elements related to the technical, operational, and compliance aspects of the project without delving into extensive details.

BIBLIOGRAPHY

- Change SSH Default Port in Linux OS Hardening By Secure SSH . YouTube. (2020a, April 23).https://youtu.be/H1gEpJ5mO_U?si=IkCogflbu7bjSr8U
- Honeypot presentation with practical demonstration. YouTube. (2020b, January 10). https://youtu.be/kVKt4t6_M-o?si=nys2_3wloeLzMRNR
- Kali Linux ssh: Learn how does SSH work in Kali linux?. EDUCBA. (2023, April 15). https://www.educba.com/kali-linux-ssh/
- Lutkevich, B., Clark, C., &amp; Cobb, M. (2021, February 24). What is a honeypot? how it protects against Cyber Attacks Security. https://www.techtarget.com/searchsecurity/definition/honey-pot
- Wikimedia Foundation. (2023, September 22). Honeypot (computing). Wikipedia. https://en.m.wikipedia.org/wiki/Honeypot_(computing).

References:

[1] Spitzner, L. (2002). Tracking hackers. Boston, MA: Addison-Wesley. Spitzner, L. (2003). The honeynet project: Trapping the hackers. IEEE Security & Privacy, 1(2),15-23.

[2] Sobesto, B., Cukier, M., Hiltune0n, M., Kormann, D., & Vesonder, G. (2011). DarkNOC: Dashboard for honeypot management. USENIX Association, 16, 16 .

[3] Weiler, N. (2002).Honeypots for distributed denial of service attacks. IEEE Computer Society,109-114.

[4] Wilson, T., Maimon, D., Sobesto, B., & Cukier, M. (2015). The effect of a surveillance banner in an attacked computer system: Additional evidence for the relevance of restrictive deterrence in cyberspace. Journal of Research in Crime and Delinquency.

[5] Wilson et al. (2015). The results show that the average number of system trespassing incidents per computer are 4.44 and 4.48 for no surveillance banner and surveillance banner computers, respectively. . Journal of Research in Crime and Delinquency.

[6] Stockman, M., Rein, R., & Heile, A. (2015). An open-source honeynet system to study system banner message effects on hackers. Journal of Computing Sciences in Colleges, pp. 282-293.

[7] Döring, C. (2005). Improving network security with honeypots. Darmstadt: University of Applied Sciences.

[8] Hoque, M. S., & Bikas, M. A. (2012). An implementation of intrusion dectection system using genetic algorithm. International Journal of Network Security & Its Applications (IJNSA).

[9] Jaiganesh, V., Sumathi, D. P., & A.Vinitha. (2013). Classification algorithms in intrusion detection system: A survey. A Vinitha et al. Int. J. Computer Technology & Applications.

[10] Stiawan, D., Abdullah, A. H., & Idris, M. Y. (2011). Characterizing network intrusion prevention system. International Journal of Computer Applications (0975–8887).