Hive Sentinel: The Hacker's Confine

By

Mariyam

B20F0165CS004

Muhammad Qaseem Ul Hassan

B20F0279CS013

Final Year Project Report

School of Computing Sciences

Pak-Austria Fachhochschule: Institute of Applied Sciences & Technology

Spring 2024

Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology

Hive Sentinel: The Hacker's Confine

A Final Year Project Report Presented to

Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology

In partial fulfillment

of the requirement for the degree of

BS Computer Science

By

Mariyam

B20F0165CS004

Muhammad Qaseem Ul Hassan

B20F0279CS013

Spring 2024

Hive Sentinel: The Hacker's Confine

A Final Year Project Report submitted to the School of Computing Sciences as partial fulfillment of the requirement for the award of BS Computer Science.

Mariyam	B20F0165CS004
Muhammad Qaseem Ul Hassan	B20F0279CS013

Academic Supervisor

Dr. Abdul Waheed Khan

Associate Professor of School of Computing Sciences

Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology

Industry Supervisor

Dr. Malik Nabeel Ahmed Awan

Head of Technology

Cydea Tech. Alpha Techno Square, Chaklala Cantt.

Rawalpindi, Punjab

Final Approval

This final year project titles

Hive Sentinel: The Hacker's Confine

By

Mariyam	*B20F0165CS004*
Muhammad Qaseem Ul Hassan	*B20F0279CS013*

under the supervision of their project supervisor and approved by the project evaluation committee, has been accepted by the Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Pakistan, in partial fulfillment of the requirements for the degree of BS Computer Science.

Academic Supervisor:

Dr. Abdul Waheed Khan
School of Computing Sciences

Industry Supervisor:

Dr. Malik Nabeel Ahmed Awan
Cydea Tech, Alpha Techno Square, Rawalpindi

HoD / Chairman:

Dr. Babr Nazir
School of Computing Sciences

<u>Declaration</u>

We, Mariyam B20F0165CS004 and Muhammad Qaseem Ul Hassan B20F0279CS013, hereby declare that we have produced the work presented in this final year project report, during the scheduled period of study. We also declare that we have not taken any material from any source except referred to wherever due to that amount of plagiarism is within an acceptable range. It is further declared that we have developed this project and the accompanied report entirely on the basis of our personal efforts made under the sincere guidance of our supervisor. No portion of the work presented in this report has been submitted in support of any other degree or qualification of this or any other University or Institute of learning, if found we shall stand responsible.

Date:

Signature:___________________
Mariyam
B20F0165CS004

Signature:___________________
Muhammad Qaseem Ul Hassan
B20F0279CS013

Certificate

It is certified that Mariyam B20F0165CS004 and Muhammad Qaseem Ul Hassan B20F0279CS013 have carried out all the work related to this project under my supervision at the School of Computing Sciences, Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology and the work fulfills the requirement for the award of BS Computer Science.

Date: ______________________

Supervisor:

____
Dr. Abdul Waheed Khan
Associate Professor

Head of Department:

____
Dr. Babar Nazir
School of Computing Sciences

DEDICATION

ACKNOWLEDGEMENTS

All praise and thanks go to Allah, the Almighty, for permitting us to take on these tasks. We would like to express our sincere gratitude to our academic supervisor, Dr. Abdul Waheed Khan, Associate Professor, School of Computing Sciences, Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, for his continuous support, guidance, and encouragement throughout this project. His expertise and insights have been invaluable in the successful completion of this work.

We also extend our heartfelt thanks to our industry supervisor, Dr. Malik Nabeel Ahmed Awan, Head of Technology at Cydea Tech, for his practical advice and support in aligning our project with industry standards and requirements.

We are grateful to our department, faculty members, and staff at the School of Computing Sciences, Pak-Austria Fachhochschule, for providing us with the necessary resources and a conducive learning environment.

Our deepest appreciation goes to our families and friends for their consistent support, patience, and understanding. Their belief in us has been a source of motivation and strength.

Lastly, we would like to thank all our colleagues and peers who have provided us with their support and feedback throughout this journey. Your contributions have been greatly valued.

ABSTRACT

In today's digital world, where businesses and individuals rely heavily on technology, cybersecurity is crucial. While firewalls and antivirus software offer basic protection, the changing tactics of cyber threats require a proactive approach. The "Hive Sentinel" project, a comprehensive solution addressing the critical issue of monitoring and managing honeypots to enhance cybersecurity. These specialized tools, strategically placed across networks, divert potential attackers, and provide invaluable insights into emerging threats. Influenced by the Deception 2.0 strategies, the project ensures proactive threat detection through features such as automated honeypot deployment, continuous health monitoring, real-time threat detection, and snapshot restoration. Moreover, the project aims to map the data to the standardized MITRE ATT&CK framework to get further insights into threat analysis.

Table of Contents

LIST OF FIGURES

LIST OF ABBREVIATIONS

ATT&CK	Adversary Tactics Techniques and Common Knowledge
SSH	Secure Shell
SMB	Server Message Block
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
URL	Uniform Resource Locater
HIH	High Interaction Honeypot
LIH	Low Interaction Honeypot
MIH	Medium Interaction Honeypot
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IOC	Indicators of Compromise
TTP	Tactics, Techniques and Procedures
APT	Advanced Persistent Threats
SOC	Security Operations Center
AV	Anti-Virus
OS	Operating System
SIEM	Security Information Event
FTP	File Transfer Protocol
ID	Identifier
IT	Information Technology
ELK	Elasticsearch, Logstash and Kibana
DBMS	Database Management System
DVWA	Damn Vulnerable Web Application
JSON	JavaScript Object Notation
TEF	Threat Event Frequency
ESXI	Elastic Sky X Integrated
XSS	Cross-Site Scripting
DSL	Domain Specific Language
CPU	Central Processing Unit
RAM	Random Access Memory
HDD	Hard Disk Drive

Chapter 1

Introduction

1. Introduction

In today's interconnected world, where businesses and individuals heavily depend on digital systems, the importance of cybersecurity cannot be overstated. As our reliance on technology grows, so does the sophistication and frequency of digital threats and cyber-attacks. Traditional security measures, such as firewalls, which act as digital barriers to control incoming and outgoing network traffic, and antivirus software, designed to detect and eliminate malicious software, form the foundation of defense against common cyber threats.

In this dynamic and evolving digital environment, actively identifying and addressing potential threats have become essential. Firewalls and antivirus tools are essential components, but they may not be sufficient on their own to tackle the ever-evolving tactics of cyber threats. Cybersecurity professionals recognize the value of a proactive approach to identify and understand the tactics, techniques, and procedures (TTPs) employed by malicious actors.

Honeypots, as a specialized cybersecurity tool, play a crucial role in this proactive defense strategy. By strategically placing decoy systems or resources across a network, organizations can lure potential attackers away from valuable assets and confidential information. These fake systems mimic genuine systems and services, inviting attackers to engage with them.

The primary objective of deploying honeypots is not just to divert and monitor unauthorized access attempts but also to gain insights into the attackers' methods and motivations. Analysis of the data collected from honeypots helps cybersecurity professionals better understand emerging threats, vulnerabilities, and attack patterns. This intelligence is invaluable for enhancing overall cybersecurity posture by enabling organizations to fine-tune their defenses, update security protocols, and develop countermeasures against evolving cyber threats.

The information gathered from honeypots contributes to threat intelligence databases, enabling the global cybersecurity community to collaborate and share knowledge about emerging cyber threats. This collective intelligence helps organizations stay ahead of cyber adversaries and fortify their defenses against potential future attacks. In essence, while traditional cybersecurity measures form a crucial baseline, honeypots add a proactive and adaptive dimension, enhancing the overall resilience of digital systems against an ever-evolving threat landscape.

1.1 Problem Statement

The lack of a dedicated health monitoring system for honeypots can lead to significant consequences for an organization's cybersecurity. Firstly, there is a risk of a false sense of security. If a honey pot is not functioning properly, it may fail to detect and capture malicious activities effectively. This can create a misleading belief that the network is adequately protected, leaving critical assets vulnerable to real threats. Secondly, the organization may face limited or inaccurate threat intelligence.

A compromised or malfunctioning honey pot may provide unreliable or incomplete information about attacker tactics and tools. This hampers the organization's ability to gather accurate threat intelligence, hindering their capacity to make informed decisions and take appropriate security measures. Furthermore, a vulnerable honey pot can introduce additional risk and exposure. Attackers can exploit the compromised honey pot as an entry point to infiltrate the network or gain insights into the organization's infrastructure. This not only undermines the effectiveness of the honey pot but also puts other systems and valuable assets at risk.

1.2 Motivation

The motivation behind the Hive Sentinel project stems from the critical need for a reliable automated and health monitoring system for honeypots in cybersecurity. Manual configuration of honeypots is often time-consuming and error-prone, leading to inconsistencies and potential security gaps. Automating the deployment of honeypots ensures uniformity, speed, and efficiency, allowing organizations to quickly set up decoy systems that are consistently configured to lure attackers effectively. Additionally, health monitoring of honeypots is crucial because when an attacker targets the system, parameters such as CPU usage, memory usage, and hard disk usage tend to spike. These indicators can signal a potential breach or an ongoing attack, enabling cybersecurity teams to respond swiftly and mitigate the threat. By continuously monitoring these health parameters, Hive Sentinel ensures that honeypots remain functional and reliable, providing accurate threat intelligence and preventing attackers from using compromised honeypots as entry points into the network.

Driven by these concerns, the Hive Sentinel project aims to enhance cybersecurity defenses by integrating automated honeypot deployment and continuous health monitoring. This proactive approach ensures that honeypots function correctly, capture accurate threat data, and prevent attackers from exploiting system vulnerabilities. By mapping the data to the MITRE ATT&CK framework and offering real-time threat detection and machine re-spawning, Hive Sentinel aims to provide comprehensive insights into emerging threats, improving the overall resilience of digital systems against sophisticated cyber-attacks.

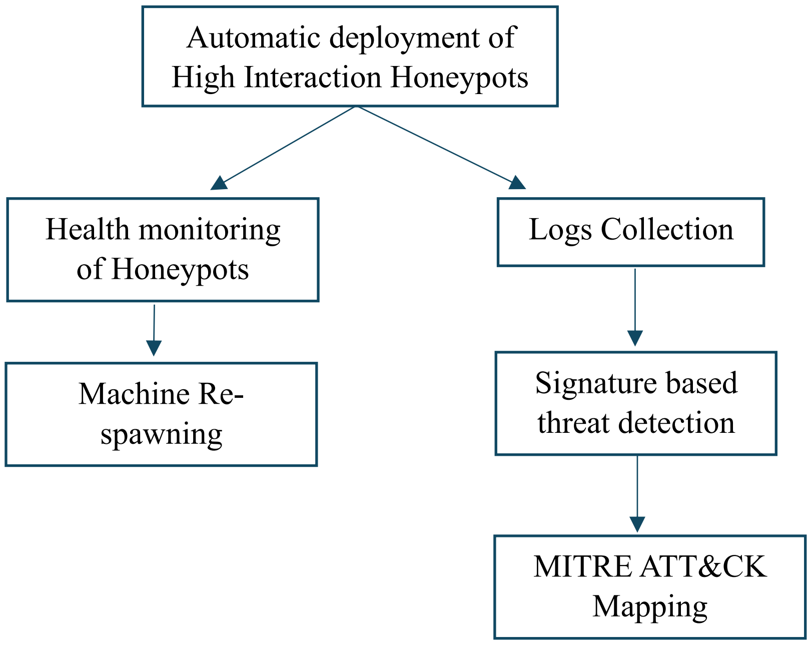
1.3 Objectives

Hive Sentinel is strategically designed to focus on advancing the capabilities of Deception 2.0 by delivering a range of advanced functionalities. Following objectives will enable the accomplishment of main aim of the project:

1. To provide automated deployment of high interaction honeypots for the deception platform 2.0.
2. To offer health monitoring for high interaction honeypots on the deception platform 2.0.
3. To provide machine re-spawning in accordance with health monitoring guidelines.
4. To enable hackers' intelligence gathering from high interaction honeypots.
5. To present a dashboard that displays gathered intelligence information through visual charts and statistics.

1.4 Scope

Among the various security mechanisms discussed—firewalls, antivirus, and honeypots—this project, “Hive Sentinel: The Hacker's Confine” specifically focuses on automating honeypot deployment and health monitoring. The basic modules of the project are discussed below as illustrated in Figure 1.



Automatic deployment of High Interaction Honeypots
Health monitoring of Honeypots
Logs Collection
Machine Re-spawning
Signature based threat detection
MITRE ATT&CK Mapping

***Figure 1**: Basic Modules of Hive Sentinel*

1) **Automatic Deployment of Honeypots:**

   Hive Sentinel places a significant emphasis on automation to enhance the efficiency and effectiveness of cybersecurity measures. The project involves the automated deployment of high interaction honeypots as part of the deception platform 2.0. These decoy systems are strategically placed to lure potential attackers, providing valuable insights into their tactics and methodologies.

2) **Logs Collection:**

   The project incorporates the essential process of log collection in honeypots, playing a vital role in detecting and analyzing malicious activities, offering insights into attack patterns, tools, and techniques for promptly identifying and containing security threats.

3) **Signature Based Threat Detection:**

   The project integrates signature-based threat detection in honeypots, using predefined patterns or signatures of known threats to identify malicious activities.

4) **MITRE ATT&Ck Mapping:**

The project involves the strategic practice of mapping generated honeypot logs to MITRE ATT&CK, providing a structured framework for understanding and categorizing cyber threats, enhancing overall threat detection capabilities.

5) **Health Monitoring:**

   The project involves a comprehensive approach to health monitoring, incorporating the collection of critical parameters such as CPU time and memory usage. This integrated health monitoring system enables the system to promptly detect any irregularities or potential issues, utilizing the extracted health monitoring parameters from each honeypot. This vigilance not only contributes to the early detection of threats but also supports the ongoing improvement of honeypot defenses.

6) **Machine re-spawning:**

   An innovative feature of the project is the machine re-spawning functionality, which operates in accordance with health monitoring guidelines. This capability ensures that compromised or malfunctioning honeypots can be restored, maintaining a resilient and continuously effective deception platform.

1.5 Literature Review

Deception 1.0 initiated the use of basic decoys and honeypots, introducing the concept of misdirection in cybersecurity. Deception 2.0 addresses the deficiencies of Deception 1.0 by offering more advanced features. This progression signifies a sophisticated understanding of attacker behavior and a commitment to staying ahead in the ever-evolving landscape of cybersecurity. A detailed overview of Deception 1.0 and Deception 2.0 is discussed below.

Deception 1.0 pioneered the use of honeypots and decoys to divert and monitor attackers, providing a foundational layer of defence. It provided valuable insights into attackers' methodologies but fell short in adaptability and automation. The demerits included static setups requiring manual intervention, limited scalability, and vulnerability to sophisticated attacks due to its lack of dynamic response mechanisms. While it helped detect threats, its effectiveness weakened against evolving cyber threats due to its static nature and reliance on easily recognizable decoys. [1]

Deception 2.0 marks a significant evolution from its predecessor, Deception 1.0, by introducing a range of advancements in the field of cyber threat mitigation. A notable improvement lies in its seamless integration of both low and high-interaction methods, offering a fluid and adaptable approach termed Hybrid Interactivity/Fluid Deception.

Unlike Deception 1.0, the new version takes customization to the next level by tailoring low-interaction services to evade emulations, thus enhancing the overall deception strategy. Scalability is efficiently addressed through a dynamic DevOps model, overcoming deployment scale challenges by automating deployment and maintenance processes, ensuring ongoing freshness, and operational efficiency.

Security measures receive a substantial boost with the introduction of deception farms and projection points, strategically placed throughout the network as virtual traps. Deception farms act as decoy fields, confusing attackers by blending real assets with fake ones, reducing the risk of security breaches. Projection points, key locations for decoys, intensify the overall deception strategy, creating a dynamic defence that disrupts attackers and makes compromising the digital environment tougher.

Moreover, the expanded deception range goes beyond traditional honeypots, allowing the system to mimic a diverse array of enterprise resources. This expansion is complemented by ecosystem integration, as Deception 2.0 actively engages with the broader security ecosystem, promoting collaboration for more effective threat detection and response.

The implementation of intrinsic threat analysis further sets Deception 2.0 apart, ensuring continuous awareness and proactive responses to emerging threats. This dynamic and sophisticated component establishes its status as a powerful tool in modern threat mitigation strategies. In comparison, Deception 1.0 may be seen as more static, lacking the integrative and adaptive features that characterize the advancements introduced in Deception 2.0. [2]

1.5.1 Key Findings

Based on the literature review, it has been established that a high-interaction honeypot must satisfy the following conditions to effectively engage with threats:

1. **Comprehensive Deception Setup:**

The honeypot must feature multiple decoys, breadcrumbs, and baits strategically installed to identify exploits used and lateral movement paths.

2. **Interconnected Decoys for Lateral Movement:**

   Lateral movement within the network should lead the attacker to other decoys, enhancing the honeypot's effectiveness.

3. **Invisible Collection Mechanism:**

   The collection mechanism must remain invisible to the attacker, ensuring covert monitoring of their activities.

4. **Network Traffic Analysis:**

   Capture and correlation of network traffic with the attacker's actions, aiding in identifying command and control centers, exfiltration activities, etc.

5. **Memory Dump Analysis:**

   Capability to review memory dumps to identify any files not saved to disk, providing a deeper understanding of the attacker's activities.

6. **Tracking Downloaded or Dropped Files:**

   Ability to track downloaded or dropped files, enabling the analysis of potentially malicious content.

Chapter 2

Background

2. Introduction

This chapter contains a comprehensive overview of honeypots, their classifications, and their pivotal role in cybersecurity through deception technology. Honeypots are systems designed to lure cybercriminals by mimicking real computer systems with actual data and processes, thereby protecting genuine networks from attacks. The chapter examines different types of honeypots low, medium, and high interaction explaining their functionalities, use cases, and their types such as Dionaea and Kippo. Additionally, it explores threat detection methodologies, contrasting signature-based and behavior-based detection approaches, and introduces the MITRE ATT&CK framework as a vital tool for understanding and counteracting adversary tactics, techniques, and procedures.

2.1 What are Honeypots?

A Honeypot is any system, file or resource that is setup to lure in adversaries, they are basically baits or trap for hackers [6].

A key aspect of honeypots is to mimic actual systems, we need to make them seem as legitimate as possible. They look like a real computer system with actual data, services and processes running that one would expect in a production environment.

They are deployed to lure or attract cybercriminals into attacking what they think is the organization’s actual network but is just a decoy, i.e., deceiving the attacker into thinking that it is a legitimate target.

By monitoring the intrusion attempts and the traffic coming into the honeypot system we can [7]:

1) Misdirect cyberattacks so our actual network is not put under risk
2) Monitor adversarial behavior and the Tactics, Techniques, and the Procedures that he is using to achieve his objectives.
3) Incorporate logging and alerting within our decoy systems to gather forensic or legal evidence to identify attackers and the systems that they are using Honeypots come under the guise of deception technology and are a key component when it comes to developing Active Defense strategies using Cyber Deception [8].

Honeypots are core part of deception technology because we are manipulating adversary perception into wasting their time and resources in exploiting vulnerabilities of these fake systems while we are also alerted of their intrusion attempts. Because these systems are only fake and are only meant to be interacted with by adversaries, any traffic or data generated by these systems is by its very nature, malicious and hostile. This in turn makes it a lot easier when it comes to identifying threats and trace out adversaries' behaviors and tactics.

Honeypots are divided into different categories depending on their purpose and level of interaction [7].

2.1.1 Low Interaction Honeypots

Low Interaction Honeypots require fewer resources. They run only a limited number of services with restricted functionality. They do not engage an attacker for very long and give basic info about the level of threat and where it is coming from.

Low Interaction Honeypots are mostly restricted to emulating different services such as SSH, SMB,HTTP etc. They give minimal access to the interact in a sense that they will not serve shells and filesystems to the attacker.

Low interaction honeypots are very easy and simple to setup. They require minimalistic configurations and are mainly useful when it comes to gaining Cyber Attribution, that is they are mainly used to track the origins of attacks, such as IP address, geolocation rather than to study attacker behavior [7], [8].

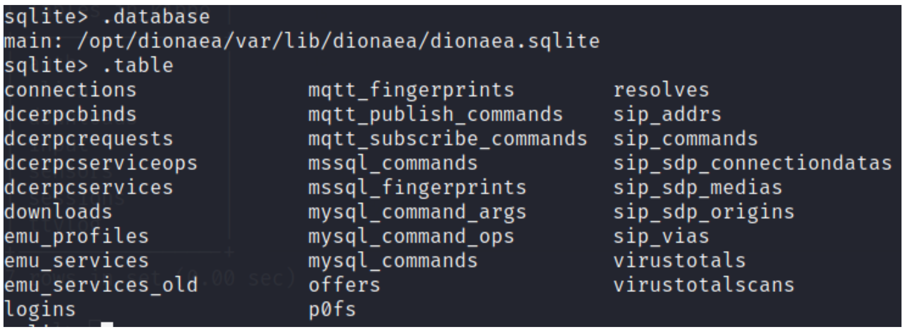
Some of the more common Low Interaction Honeypots are:

2.1.1.1 Dionaea

Dionaea, a successor of Nepenthes, is a low interaction honeypot whose main purpose is to trap malware and to capture malware binaries [9].

Dionaea emulates a number of services and protocols with known vulnerabilities. Dionaea has a mechanism of incidents and incident handlers. Incidents include information about the attack type, time and date of the attack and the incident handlers are responsible for logging those incidents and evaluating the payloads captured by Dionaea honeypot.

Dionaea uses an SQLite database as its main logging database. Figure 2 shows the tables that are part of this database.



```
sqlite> .database
main: /opt/dionaea/var/lib/dionaea/dionaea.sqlite
sqlite> .table
connections              mqtt_fingerprints        resolves
dcerpcbinds              mqtt_publish_commands    sip_addrs
dcerpcrequests           mqtt_subscribe_commands  sip_commands
dcerpcserviceops         mssql_commands           sip_sdp_connectiondatas
dcerpcservices           mssql_fingerprints       sip_sdp_medias
downloads                mysql_command_args       sip_sdp_origins
emu_profiles             mysql_command_ops        sip_vias
emu_services             mysql_commands           virustotals
emu_services_old         offers                   virustotalscans
logins                   p0fs
```

Figure 2: *Dionaea SQLite Database*

Dionaea also supports a front-end web interface to better visualize the generated logs [9].

Figure 3 shows the different metrics visualized using the frontend.

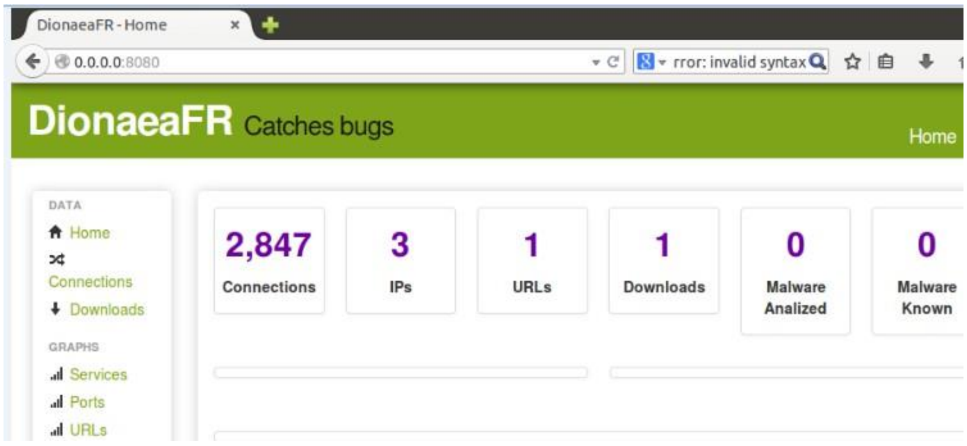


Figure 3: *Dionaea Frontend*

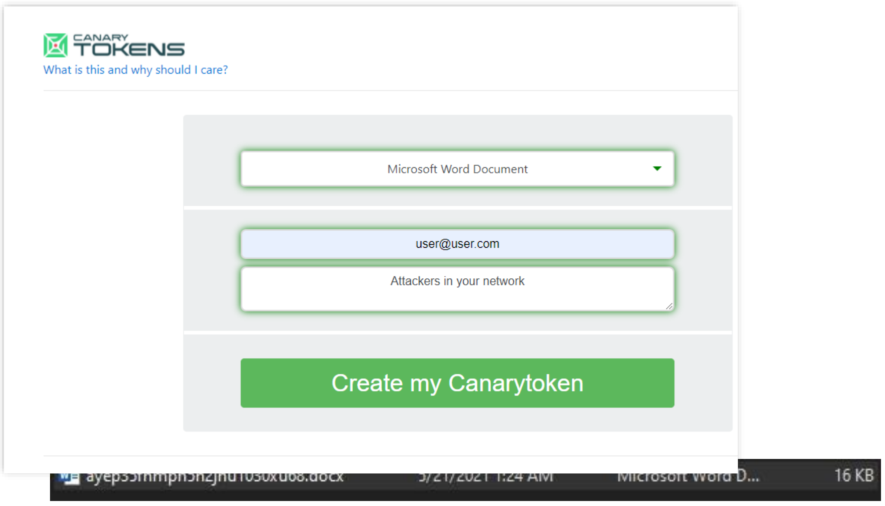
2.1.1.2 CanaryTokens

Canary Tokens are a special type of deception technology called Honeytokens, which can consist of HoneyDocs, HoneyURLS etc.

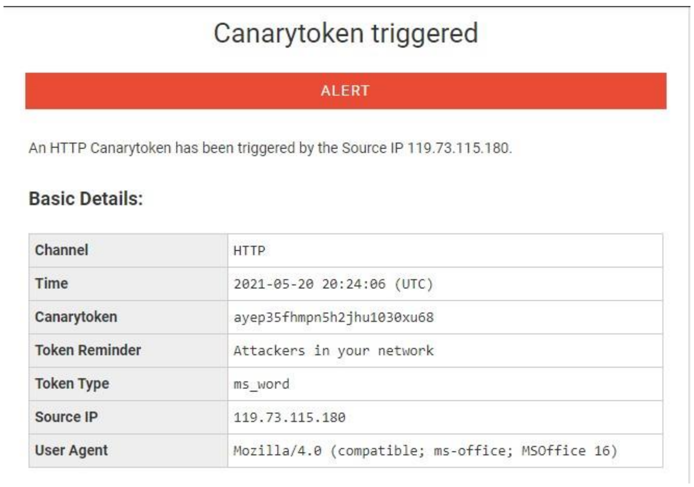
Using Canary Tokens, we can create different Honeytokens that have beaconing and alerting capabilities.

This is a cyber deception technique, and we can make it effective by making it seem as legitimate as possible. Consider a HoneyDoc with a name that would attract an intruder (like passwords.docx), the attacker will be tempted to access this file and when he does, we will be alerted and gain attribution of the attacker such as IP address and rough geolocation.

Many tools are also available we can use to create our own honeytokens and do more than just getting basic information about the system. We can include code and batch scripts that execute certain processes on the compromised system. Figure 4 shows the creation of a honeydoc, and consequent attribution gained from a it after accessing the canarytoken.


CANARY TOKENS
What is this and why should I care?
Microsoft Word Document
user@user.com
Attackers in your network
Create my Canarytoken
ayep35fhmpn5h2jhu1030xu68.docx 5/21/2021 1:24 AM Microsoft Word D... 16 KB

#


Canarytoken triggered

ALERT

An HTTP Canarytoken has been triggered by the Source IP 119.73.115.180.

Basic Details:

Channel	HTTP
Time	2021-05-20 20:24:06 (UTC)
Canarytoken	ayep35fhmpn5h2jhu1030xu68
Token Reminder	Attackers in your network
Token Type	ms_word
Source IP	119.73.115.180
User Agent	Mozilla/4.0 (compatible; ms-office; MSOffice 16)

Figure 4: *Attribution From Honey Tokens (Canarytoken)*

2.1.2 Medium Interaction Honeypot (MIH)

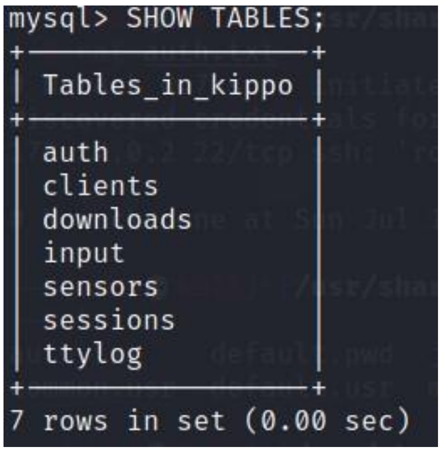
A medium interaction honeypot is a middle ground between low interaction and high interaction honeypot technology. They are mainly used in cases where certain attribution of the attacker is required, however large-scale attacks are not as in their high interaction counterparts.

Medium interaction honeypots have the advantage of providing a higher level of interaction whilst still being fairly lightweight and simple to setup.

2.1.2.1 Kippo

Kippo is a medium to high interaction honeypot depending on how it is setup and configured. Kippo emulates SSH and telnet services [11]. Kippo has the capability of offering a shell to the attacker with an entire fake filesystem on successful authentication. Kippo is mainly used to log brute force authentication attempts and the shell interactions of the attacker. Kippo uses MySQL as its logging database.

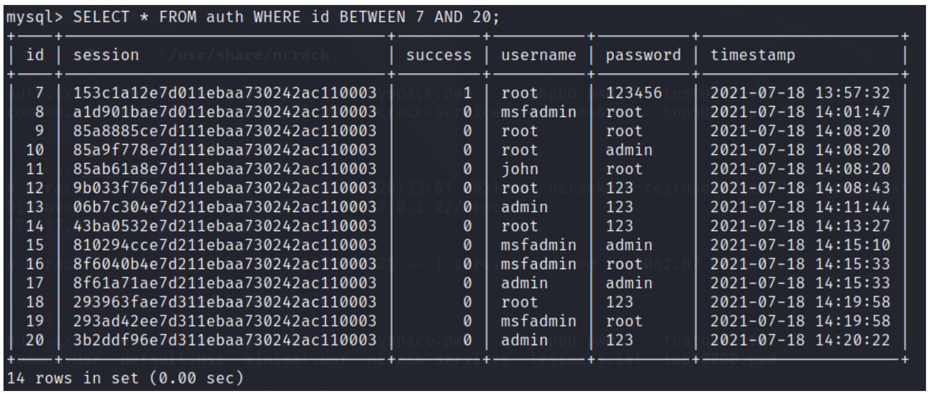
Figure 5 shows the tables that are part of this database.



```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_kippo |
+-----------------+
| auth            |
| clients         |
| downloads       |
| input           |
| sensors         |
| sessions        |
| ttylog          |
+-----------------+
7 rows in set (0.00 sec)
```

Figure 5: *Kippo MYSQL Database*

A sample of authentication attempts (successful and unsuccessful) stored by Kippo are shown below in **Figure 6**



```
mysql> SELECT * FROM auth WHERE id BETWEEN 7 AND 20;
+----+----------------------------------+---------+----------+----------+---------------------+
| id | session                          | success | username | password | timestamp           |
+----+----------------------------------+---------+----------+----------+---------------------+
|  7 | 153c1a12e7d011ebaa730242ac110003 |       1 | root     | 123456   | 2021-07-18 13:57:32 |
|  8 | a1d901bae7d011ebaa730242ac110003 |       0 | msfadmin | root     | 2021-07-18 14:01:47 |
|  9 | 85a8885ce7d111ebaa730242ac110003 |       0 | root     | root     | 2021-07-18 14:08:20 |
| 10 | 85a9f778e7d111ebaa730242ac110003 |       0 | root     | admin    | 2021-07-18 14:08:20 |
| 11 | 85ab61a8e7d111ebaa730242ac110003 |       0 | john     | root     | 2021-07-18 14:08:20 |
| 12 | 9b033f76e7d111ebaa730242ac110003 |       0 | root     | 123      | 2021-07-18 14:08:43 |
| 13 | 06b7c304e7d211ebaa730242ac110003 |       0 | admin    | 123      | 2021-07-18 14:11:44 |
| 14 | 43ba0532e7d211ebaa730242ac110003 |       0 | root     | 123      | 2021-07-18 14:13:27 |
| 15 | 810294cce7d211ebaa730242ac110003 |       0 | msfadmin | admin    | 2021-07-18 14:15:10 |
| 16 | 8f6040b4e7d211ebaa730242ac110003 |       0 | msfadmin | root     | 2021-07-18 14:15:33 |
| 17 | 8f61a71ae7d211ebaa730242ac110003 |       0 | admin    | admin    | 2021-07-18 14:15:33 |
| 18 | 293963fae7d311ebaa730242ac110003 |       0 | root     | 123      | 2021-07-18 14:19:58 |
| 19 | 293ad42ee7d311ebaa730242ac110003 |       0 | msfadmin | root     | 2021-07-18 14:19:58 |
| 20 | 3b2ddf96e7d311ebaa730242ac110003 |       0 | admin    | 123      | 2021-07-18 14:20:22 |
+----+----------------------------------+---------+----------+----------+---------------------+
14 rows in set (0.00 sec)
```

Figure 6: *Kippo Auth Table showing authentication attempts*

Kippo also comes along with a front-end web interface called Kippo Graph, shown in **Figure 7**, which can be used to view the data logged by Kippo in a graphical format by generating pie charts and bar graphs [11].

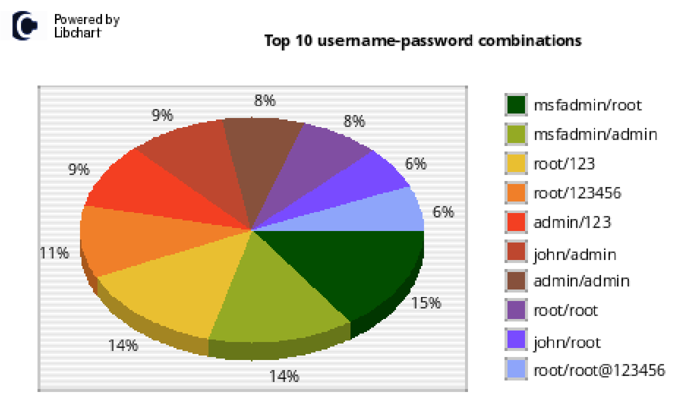


Figure 7: *Data Visualization using Kippo Graph*

2.1.3 High Interaction Honeypots (HIH)

High Interaction Honeypots are resource hungry and require constant monitoring and maintenance. HIH consist of a whole decoy computer system with files, applications and different running processes and services. High Interaction Honeypots engage an attacker for a significant amount of time and gives extensive and in-depth information about adversarial behavior. Instead of simply simulating specific protocols or services, the attacker is given real systems to attack, making it far less likely that they will suspect they are being diverted or observed [11], [12].

When setting up a high interaction honeypot, special precautions need to be taken in order to limit the reign the attacker has on the system, so that he does not use it to launch attacks on other systems in the network. They must be allowed to breach the machine and engage in some activity without being able to utilize their control of the system to exploit legitimate systems on the network [10].

Proper logging is an essential part of a High Interaction Honeypot as the attacker is interacting with a full-fledged system, and since there is high interactivity, consequently there are more components that need to be monitored.

Typically following resources should be monitored in a High Interaction honeypot [13]:

1. Files
2. Registry (in case of Windows systems)
3. Processes
4. Network Traffic
5. CPU Health

As there is a wide variety of data collected from such honeypots, this allows for the investigation of the attacker's methods and tactics for gaining additional access or performing malicious tasks.

2.1.4 Research Honeypots

Research Honeypots are mainly used to study adversarial behavior and get a better understanding of the tools and techniques they utilize. Research Honeypots give us an insight into attacker's motives and objectives that he is trying to achieve and how they achieve them.

Research honeypots are mainly setup for educational value and not with the intent of protecting live networks and systems. Information gained from such honeypots is vital in understanding attacker trends and patterns, and in turn help in developing security tools and better defensive strategies to counteract those adversarial trends.

2.1.5 Production Honeypots

Production Honeypots are mainly deployed alongside a production environment. These honeypots are primarily used to detect and misdirect cyber intrusions, so the actual networks of the organization are not targeted.

These are deployed to actively defend systems and networks in real time. These types of honeypots are becoming more popular as they go well with intrusion detection systems and current security measures. The level of interactivity and logging that these honeypots provide can vary from case to case.

Production honeypots are popular among businesses because they are simple to deploy while exposing critical information about cyber threats and vulnerabilities that threaten their networks.

2.2 Threat/Intrusion Detection Postures

The method of examining the complete security ecosystem to discover any malicious behavior that could compromise the network is known as threat detection. If a threat is found, mitigating measures must be taken to effectively neutralize the threat before it can exploit any existing vulnerabilities.

A hardware or software program that is used to monitor a network or a system for possible malicious activities and intrusion attempts is known as an Intrusion Detection System (IDS). An IDS is continuously on the lookout for possible attack vectors and malware, and triggers alerts when any such malicious activity is found.

Owing to this there are mainly two types of threat detection approaches in cybersecurity.

2.2.1 Signature Based Threat Detection

The "attack signature" was originally used by antivirus makers to scan system files for signs of malicious activity. A signature-based intrusion detection system (IDS) typically examines inbound network data for sequences and patterns that match a certain attack signature. These can be identified in network packet headers as well as data sequences matching recognized malware or other dangerous patterns. An attack signature can also be identified in destination or source network addresses, as well as specific data sequences or packet series.

Signature-based detection employs a predefined set of Indicators Of Compromise (IOCs). Specific network attack behaviors, recognized byte sequences, and malicious domains are examples of these. Email topic lines and file hashes may also be included. These are essentially footprints left behind by a malware.

One of the most significant shortcomings of signature-based intrusion detection systems is their inability to detect unknown threats [12]. To avoid detection, malicious actors might simply alter their attack sequences within malware and other sorts of attacks. Traffic can also be encrypted to totally avoid detection by signature-based methods. Furthermore, APTs typically involve threat actors who modify their signature more than 60% of the time [12].

Almost all traditional Anti-Virus Technology work on this approach.

2.2.2 Behavior Based Threat Detection

A behavior or anomaly-based IDS system detects and analyses harmful or unusual patterns of activity in addition to identifying specific attack signatures. This type of technology uses statistical, AI, and machine learning techniques to analyze massive amounts of data and network traffic and identify anomalies [3], [4], [5].

Rather than looking for patterns associated with certain types of attacks, behavior-based intrusion detection systems (IDS) monitor behaviors that may be associated with threats, increasing the possibility of detecting and mitigating a hostile action before the network is penetrated [13].

Behavior-based IDS solutions provide the finest line of protection against network breaches by intelligently analyzing data using AI and machine learning. They offer comprehensive perspectives of today's complicated, expansive networks. This means that malicious and unusual traffic will be identified throughout the full physical and virtual network attack surfaces.

As it comes down, signature-based detection approach is very useful when it comes to detecting knows threat, however, is unable to detect novel attacks and malware. Behavior based approach is very likely to detect novel and unknown attacks and malware as it shortlists threats by monitoring for behavior rather that IOCs [12], [13].

2.3 MITRE ATT&CK Framework

MITRE ATT&CK® is an acronym that stands for MITRE Adversary Tactics, Techniques, and Common Knowledge (ATT&CK). The MITRE ATT&CK framework is a curated knowledge base and model for cyber adversary behavior that reflects the many stages of an adversary's attack lifecycle as well as the platforms that they are known to target. The model's abstraction of tactics and approaches provides a standard taxonomy of specific adversary acts understood by both the offensive and defensive sides of cybersecurity. It also offers an adequate level of categorization for adversary conduct as well as precise techniques to defend against it [14].

MITRE ATT&CK® is a worldwide knowledge base based on real-world observations of enemy tactics, techniques, and procedures. In the business sector, government, and the cybersecurity product and service community, the ATT&CK knowledge base is utilized as a foundation for creating specialized threat models and approaches.

MITRE ATT&CK's behavioral model consists of the following main components:

1. **Tactics:** which define the short-term goals of the adversary during an attack (columns in the matrix)
2. **Techniques:** these are the ways by which attackers achieve their short-term goals (cells in the matrix)
3. **Procedures:** These are the documented uses of the techniques by different adversary groups also called Advanced Persistent Threats (APTs) that have been observed in the wild.

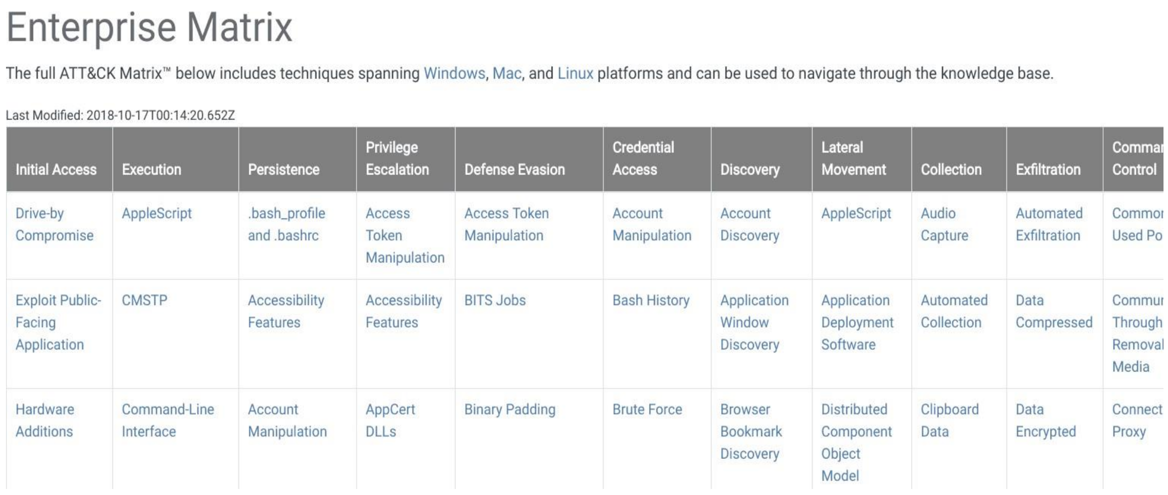
The MITRE ATT&CK matrix contains a collection of strategies used by adversaries to achieve a certain goal. In the ATT&CK Matrix, these objectives are classified as tactics. The objectives are given in a sequential fashion, beginning with reconnaissance, and ending with exfiltration or impact.

Figure 8 shows a portion of the adversarial tactics that are part of MITRE ATT&CK which include:

1. **Reconnaissance:** involves gathering intel and information about target
2. **Resource Development:** setting up a command-and-control center
3. **Initial Access:** involves techniques which are used to compromise systems in the initial stages. i.e., spear phishing.
4. **Execution:** executing payload, malware on target system
5. **Persistence:** trying to maintain foothold in the target.
6. **Privilege Escalation:** trying to gain higher-level permissions.
7. **Defense Evasion:** trying to avoid being detected.
8. **Credential Access:** stealing accounts names and passwords.
9. **Discovery:** trying to figure out your network or domain.
10. **Lateral Movement:** moving through your environment
11. **Collection:** gathering data of interest to the adversary goal
12. **Command and Control:** communicating with compromised systems to control them
13. **Exfiltration:** stealing (exfiltrating) data
14. **Impact:** manipulate, interrupt, or destroy systems and data

Under each tactic we have different techniques, some of which are shown in Figure 8. These techniques may even have sub-techniques that provide further detail on how an adversary propagates with his attack [14].

A portion of the Enterprise ATT&CK Matrix is show below in Figure 8:



Enterprise Matrix

The full ATT&CK Matrix™ below includes techniques spanning Windows, Mac, and Linux platforms and can be used to navigate through the knowledge base.

Last Modified: 2018-10-17T00:14:20.652Z

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Comma Control
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Automated Exfiltration	Commor Used Po
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	BITS Jobs	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Data Compressed	Commur Through Removal Media
Hardware Additions	Command-Line Interface	Account Manipulation	AppCert DLLs	Binary Padding	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Data Encrypted	Connect Proxy

Figure 8: *ATT&CK Matrix enumerating attacker TTPs*

2.3.1 ATT&CK vs Cyber Kill Chain

The Lockheed Martin Cyber Kill Chain® (shown in Figure 9) is another framework and resource that is used to model how an adversary propagates his attack through a network. It defines different phases during the attack path of an adversary, the attack chain consists of the following phases as shown in the figure.

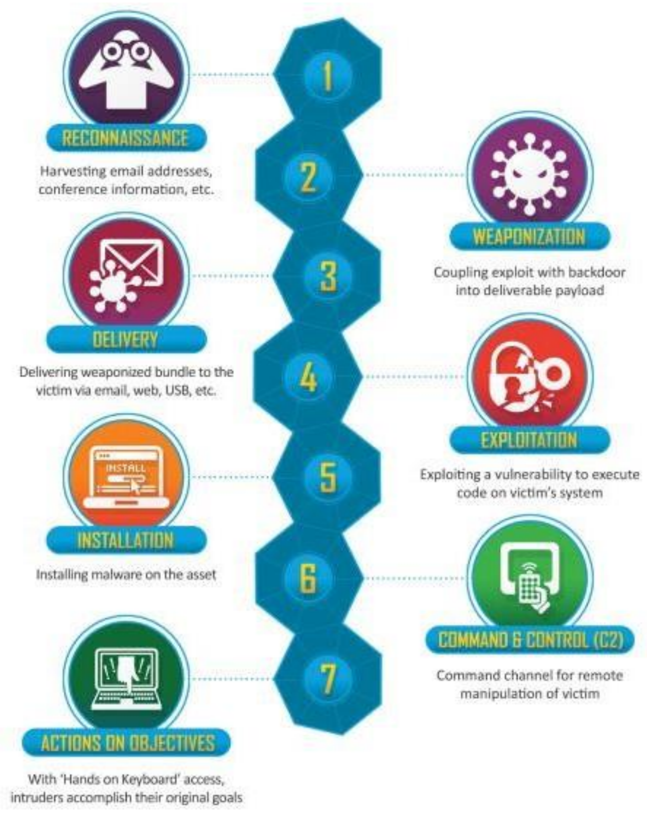

1
RECONNAISSANCE
Harvesting email addresses, conference information, etc.
2
WEAPONIZATION
Coupling exploit with backdoor into deliverable payload
3
DELIVERY
Delivering weaponized bundle to the victim via email, web, USB, etc.
4
EXPLOITATION
Exploiting a vulnerability to execute code on victim's system
5
INSTALL
INSTALLATION
Installing malware on the asset
6
COMMAND & CONTROL (C2)
Command channel for remote manipulation of victim
7
ACTIONS ON OBJECTIVES
With 'Hands on Keyboard' access, intruders accomplish their original goals

Figure 9 *Cyber Kill Chain*

The overarching attacker objectives in each attack phase are almost similar in both frameworks. However, the key difference is that the ATT&CK Matrix goes into much more depth on how each phase of the attack is conducted by also listing out the techniques (Figure 9). Other than that, the ATT&CK framework is also regularly updated.

2.3.2 What Can be Done With ATT&CK

There are different ways in which organizations and Security Operation Centers (SOCs) can utilize the ATT&CK Matrix [13], [14]:

1. **Adversary Emulation:** Evaluate defensive strategies and security postures by emulating attacker behavior. ATT&CK can be used to create adversary emulation scenarios to test and verify defenses.
2. **Red Teaming:** Act as the attacker to provide security feedback after a breach. ATT&CK can be used to plan, create, execute, and report red team operations.
3. **Behavioral Analytics Development:** involves analyzing suspicious activity to ascertain whether it is malicious or not.
4. **Defensive Gap Analysis:** ATT&CK can be used to determine where the defenses of an organization lag (gap). Constantly evaluating and analyzing our security infrastructure to find gaps and loopholes that could be exploited.
5. **Threat Hunting:** proactively looking for threat in your networks and systems. MITRE provides us with a baseline of what to look for.

2.3.3 TP Based Threat Hunting Using MITRE

TTP stand for Tactics, Techniques and Procedures and theses define the Why and How of the attackers. TTP based threat detection comes under the guise of behavior-based threat detection. The TTPs define the behavior of an attacker within a system.

TTP based detection involves characterizing attacker behavior and techniques that attackers utilize to achieve their objectives. The MITRE ATT&CK Matrix is an adequate tool to characterize those strategies [13].

Chapter 3

Software Requirements Specifications (SRS)

3. Introduction

In software development, the Software Requirements Specification (SRS) document is like a guide, giving a detailed plan for the entire project. It's crucial for everyone involved—clients, developers, and testers—to understand what the software is meant to do.

The SRS document is essential because it acts as a roadmap for the development team throughout the project. It helps communication between clients and developers and serves as a reference point for the project's different stages. A well-done SRS reduces misunderstandings, lowers project risks, and sets the stage for successful software.

This chapter encompasses critical components fundamental to the project's success, providing insights into objectives, framework and methodology, functional requirements, hardware specifications, proposed tools, platforms, and the anticipated project timeline.

3.1 Functional Requirements

The project must meet the following functional requirements to establish an effective honeypot system that will efficiently contribute to proactive cybersecurity defense mechanisms.

1. A dashboard providing the following functionalities:
   - ➢ Automatically create virtual machines and deploy vulnerable services on them.
   - ➢ Monitor health parameters of honeypots and facilitate machine re-spawning.
   - ➢ Analyze MITRE ATT&CK techniques.
   - ➢ Control the management of honeypots.
2. Continuous monitoring of system resources, network connectivity, and application services of honeypots.
3. Real-time detection and alerting of suspicious or malicious activities targeting the honeypots.
4. Generation of detailed logs, reports, and analysis of honeypot activities for threat intelligence purposes.

5. Mapping of generated logs with MITRE ATT&CK to get insights into hacker tactics, techniques, and procedures.
6. Monitor and organize security events in one central place.
7. Control management system to monitor health of honeypots.
8. Snapshot restoration capabilities to reset honeypots to a clean state after attacks or specific penetration levels.

3.2 Hardware Specification

1. **Server: Dell PowerEdge R740**

   In the Hive Sentinel project, a server is essential for creating multiple virtual machines, including honeypots and real machines. The server has the capability to run 24/7, aligning with the project's requirement for continuous honeypot operation to protect the real system. The Dell PowerEdge R740 server fulfills the project needs.

   The Dell PowerEdge R740 is a high-performance server known for its reliability and scalability. Its robust architecture and management capabilities make it an ideal choice for handling complex tasks like control management and health monitoring in honeypot projects. It offers features for remote management, ensuring efficient control and monitoring of honeypots spread across different environments.

2. **RAM: 64 GB**

   The 64 GB RAM capacity is crucial for this project as it allows for seamless multitasking and data handling. In a honeypot environment, where multiple instances might run simultaneously, this ample RAM ensures that the system can process and analyze a significant amount of data without experiencing performance bottlenecks. It supports concurrent processes for real-time monitoring and analysis of activities within the honeypots.

3. **CPU Cores: 8**

   In this project, the utilization of 8 CPU cores is essential to leverage substantial processing power. These cores enable efficient parallel processing, allowing for quick analysis of incoming data, rapid responses to potential threats, and the ability to run multiple monitoring and analysis tools concurrently. This capability is vital for ensuring timely threat detection and response in a dynamic environment.

4. **Storage: 1 TB**

   In this project, a 1 TB storage capacity is essential for storing logs, captured data, and information collected from honeypots. In a security monitoring scenario, where vast amounts of data are generated regularly, having sufficient storage is critical for retaining historical data for forensic analysis, identifying patterns, and investigating security incidents. It provides the necessary space to archive logs and maintain a comprehensive record of honeypot activities.

   These specifications collectively form the foundation for an efficient and robust infrastructure, capable of handling the diverse demands of control management and health monitoring within a honeypot project. They ensure that the system remains responsive, capable of handling extensive data processing, and equipped to support the project's objectives of security analysis and threat detection.

3.3 Proposed Tools and Platforms

3.3.1 Sysmon

System Monitor (Sysmon) is a Windows-based device driver and system service which keeps track of all system activities even after a system reboot and logs it to the Windows event log. It augments the regular Windows logs with higher-level monitoring of events. It gives detailed information on the development of processes, network connections, and changes to the time taken to create files in a repository. Sysmon logs will be collected and exported to a centralized database for analysis, aiming to detect anomalous and potentially malicious activities.

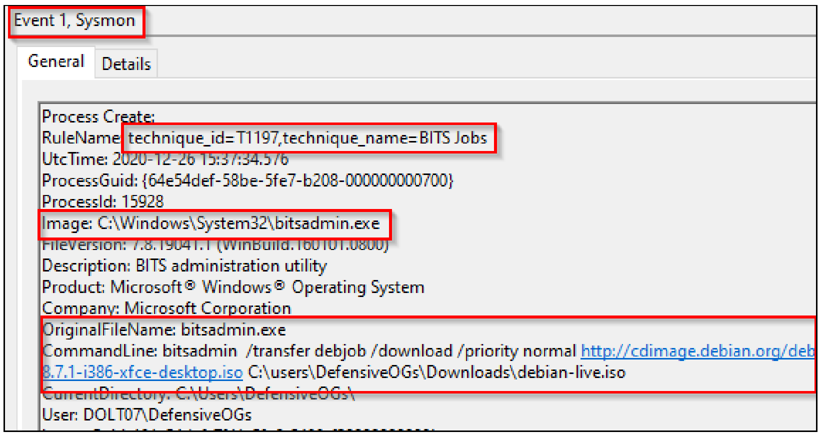
1. Sysmon has the following capabilities:
2. Logs process creation with complete command line
3. Logs file hashes (MD5, IMPASH, SHA256)

4. Process GUID (Global User ID) to correlate process events where IDs are reused.
5. Session GUID to keep track of events and logs generated during a logon session
6. Also logs network connection events, including Source and Destination IP/Port.
7. Capable of logging modifications to file timestamps, a common technique utilized by malware for defence evasion purposes.
8. Filtering rules to include or exclude certain events

Sysmon logs are stored under its own channel under "Applications and Services Logs" in the standard Windows Event Viewer. Sysmon monitors and logs events based on Event ID, where each event is assigned a unique ID. The configuration file for Sysmon, typically named "sysmonconfig.xml," plays a crucial role in determining which events are monitored and logged. This XML-based configuration file allows users to customize Sysmon's behaviour by specifying rules and filters for different types of events. By specifying customized rules in Sysmon configuration file we can create signature-based detection.

3.3.1.1 Event ID 1-Process Creation

The process creation event provides extended information about a newly created process. The full command line provides context on the process execution. Figure 10 shows a sample of Event ID 1 captured by Sysmon and viewed in the Windows Event Logs. We can see the different parameters logged by Sysmon such as Image, ParentID, CommandLine, etc.



```
Process Create:
RuleName technique_id=T1197,technique_name=BITS Jobs
UtcTime: 2020-12-26 15:37:34.576
ProcessGuid: {64e54def-58be-5fe7-b208-000000000700}
ProcessId: 15928
Image: C:\Windows\System32\bitsadmin.exe
FileVersion: 7.8.19041.1 (WinBuild.160101.0800)
Description: BITS administration utility
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: bitsadmin.exe
CommandLine: bitsadmin  /transfer debjob /download /priority normal http://cdimage.debian.org/deb
8.7.1-i386-xfce-desktop.iso C:\users\DefensiveOGs\Downloads\debian-live.iso
CurrentDirectory: C:\Users\DefensiveOGs\
User: DOLT07\DefensiveOGs
```

Figure 10: *Event ID 1 Process Create logged using Sysmon*

3.3.1.2 Event ID 3-Network Connection

The Network Connection Event logs all processes establishing network connection along with the source and destination IP/Port. Figure 11 shows a sample of Event ID 3 captured by Sysmon and viewed in the Windows Event Logs with some of the important attributes highlighted.

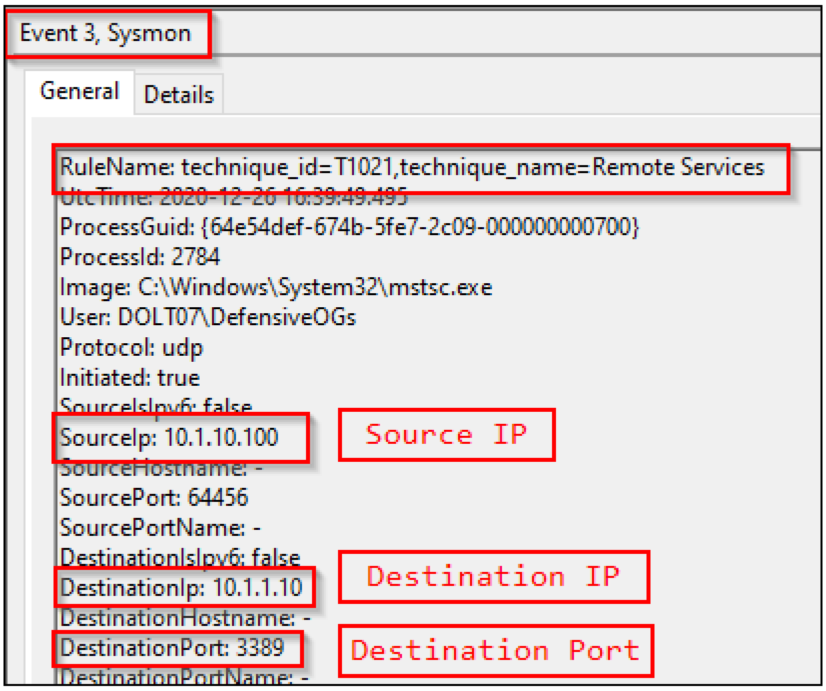
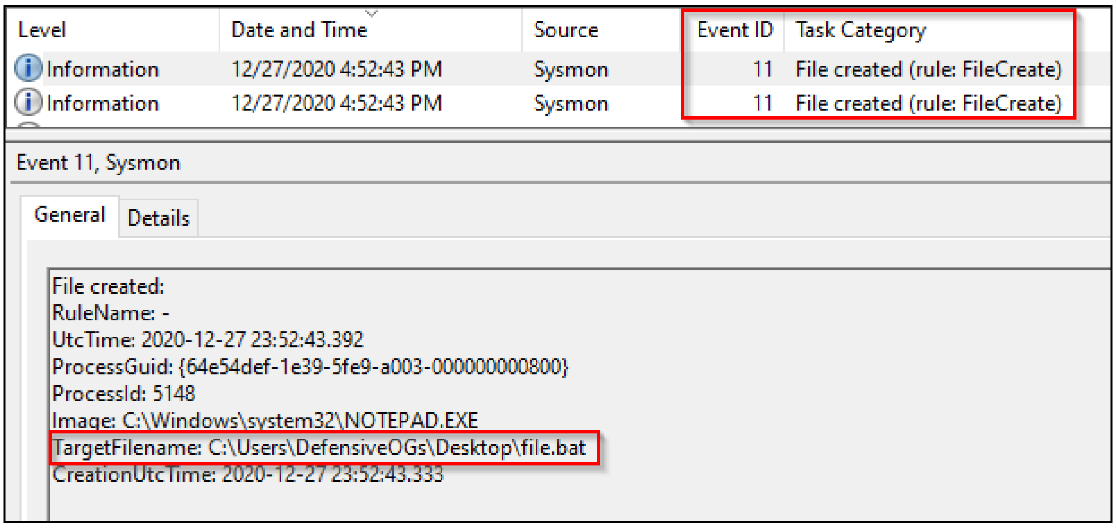

```
Event 3, Sysmon
General  Details
RuleName: technique_id=T1021,technique_name=Remote Services
UtcTime: 2020-12-26 16:39:49.495
ProcessGuid: {64e54def-674b-5fe7-2c09-000000000700}
ProcessId: 2784
Image: C:\Windows\System32\mstsc.exe
User: DOLT07\DefensiveOGs
Protocol: udp
Initiated: true
SourceIsIpv6: false
SourceIp: 10.1.10.100
SourceHostname: -
SourcePort: 64456
SourcePortName: -
DestinationIsIpv6: false
DestinationIp: 10.1.1.10
DestinationHostname: -
DestinationPort: 3389
DestinationPortName: -
```

Figure 11: *Event ID 3 logged using Sysmon*

4.3.1.3 Event ID 11: File Create

These events are logged whenever a new file is created, or an existing file is overwritten. By default, these events can be very verbose and require some filtering. Figure 12 shows a sample of Event ID 15 captured by Sysmon and viewed in the Windows Event Logs with some of the important attributes highlighted, which include Image (the process creating the file), TargetFilename (the path of the file created) etc.

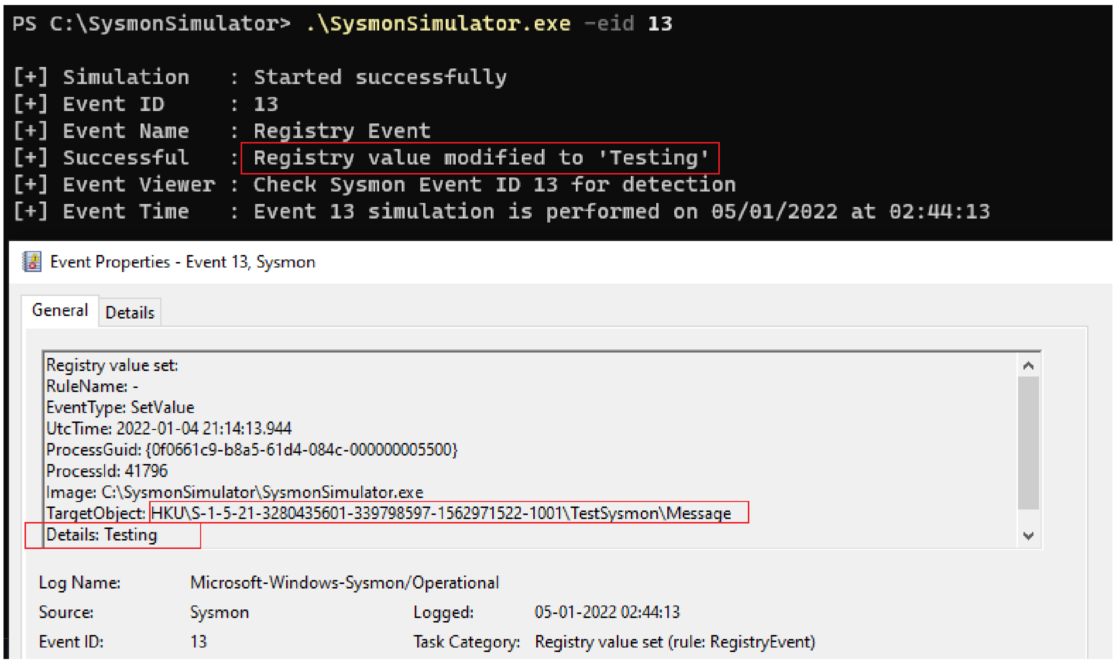


```
Level          Date and Time            Source    Event ID  Task Category
Information    12/27/2020 4:52:43 PM    Sysmon         11   File created (rule: FileCreate)
Information    12/27/2020 4:52:43 PM    Sysmon         11   File created (rule: FileCreate)

Event 11, Sysmon
General  Details

File created:
RuleName: -
UtcTime: 2020-12-27 23:52:43.392
ProcessGuid: {64e54def-1e39-5fe9-a003-000000000800}
ProcessId: 5148
Image: C:\Windows\system32\NOTEPAD.EXE
TargetFilename: C:\Users\DefensiveOGs\Desktop\file.bat
CreationUtcTime: 2020-12-27 23:52:43.333
```

***Figure 12** Event ID 11 logged using Sysmon*

3.3.1.4 Event ID 12-14: Registry Events

All events regarding registry modification, registry object creation/deletion, key/value creation deletion, and value set are logged under Event ID12-14 respectively a sample of such and event logged is shown in Figure 13 below with the relevant fields.



```
PS C:\SysmonSimulator> .\SysmonSimulator.exe -eid 13

[+] Simulation    : Started successfully
[+] Event ID      : 13
[+] Event Name    : Registry Event
[+] Successful    : Registry value modified to 'Testing'
[+] Event Viewer  : Check Sysmon Event ID 13 for detection
[+] Event Time    : Event 13 simulation is performed on 05/01/2022 at 02:44:13
```

```
Event Properties - Event 13, Sysmon
General  Details

Registry value set:
RuleName: -
EventType: SetValue
UtcTime: 2022-01-04 21:14:13.944
ProcessGuid: {0f0661c9-b8a5-61d4-084c-000000005500}
ProcessId: 41796
Image: C:\SysmonSimulator\SysmonSimulator.exe
TargetObject: HKU\S-1-5-21-3280435601-339798597-1562971522-1001\TestSysmon\Message
Details: Testing

Log Name:    Microsoft-Windows-Sysmon/Operational
Source:      Sysmon                 Logged:         05-01-2022 02:44:13
Event ID:    13                     Task Category:  Registry value set (rule: RegistryEvent)
```

***Figure 13:** Event ID 13 logged using sysmon*

The capability of Sysmon of logging certain events under fixed event ids makes the process of collecting relevant logs and analyzing them a lot easier [19].

3.4 ELK Stack

ELK Stack is the most popular open-source IT log management solution for businesses that desire the benefits of a centralized logging system without the cost of enterprise software. The ELK Stack comprised three open-source products - Elasticsearch, Logstash, and Kibana - developed, managed, and maintained by Elastic until around a year ago. Beats' introduction and later inclusion transformed the stack into a project with four legs. Logs from all your systems and apps may be aggregated into an Elasticsearch database, which can then be analyzed for speedier troubleshooting or security analytics and provide visualizations for application and infrastructure monitoring [18].

Apache Lucene is the foundation for Elasticsearch, a distributed search and analytics engine. When Elasticsearch was first released in 2010, it immediately became the most widely used search engine for various purposes, including log analysis, full-text search, and the gathering of security, business, and operational information.

Logstash, integrated with the ELK Stack, collects data from several sources, transforms it on the fly, and delivers it to your intended destination using an open-source, server-side data pipeline. Logstash is a popular choice for loading data into Elasticsearch because of its tight interaction with Elasticsearch, powerful log processing features, and over 200 pre-built opensource plugins that may help you easily index your data.

A "data shipper" known as "Beat" is also integrated with ELK Stack and used to deliver operational data to Elasticsearch. This data can be upgraded or archived in Logstash.

When it comes to displaying data stored in Elasticsearch, Kibana is an obvious choice because of its close connection with Elasticsearch itself. Log and time-series analytics, application monitoring, and operational intelligence are all examples of use cases for Kibana. Powerful and user-friendly tools like histograms, pie charts, and heat maps are included as built-in geographic support.

Elasticsearch is a distributed, open-source search and analytics engine known for its scalability and real-time search capabilities. It efficiently stores and indexes large volumes of data, accommodating complex queries and full-text searches. Elasticsearch enables centralized monitoring. By integrating Hive Sentinel with Elasticsearch, data from diverse machines are collected and combined in a centralized repository, enhancing the overall efficiency of analysis and insights.

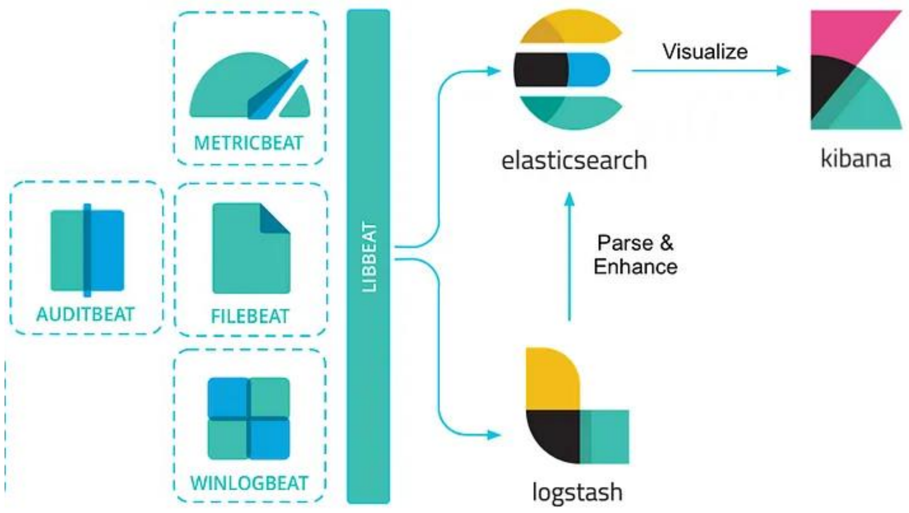


Figure 14: *Workflow of ELK*

3.4.1 Beats of ELK

Beats are lightweight data shippers designed to send data from hundreds or thousands of machines and systems to Elasticsearch or Logstash. They are part of the Elastic Stack (formerly known as ELK Stack), which includes Elasticsearch, Logstash, and Kibana. Beats collect and forward various types of data, including logs, metrics, and network data.

Popular types of Beats include:

Filebeat: Collects and ships log files. It is commonly used for log aggregation.

Metricbeat: Collects and ships system and service metrics. It monitors CPU, memory, and other system metrics.

Winlogbeat: Collects and ships Windows Event logs. It is used for monitoring Windows environments.

Auditbeat: Collects audit data from the host, providing insights into system activities and changes.

3.5 Damn Vulnerable Web Application

Online application DVWA is a PHP/MySQL-based web application that aims to let security experts evaluate their abilities and products in a legal setting. Its primary goal is to assist security professionals in testing their strategies and training in a legal structure, assist software developers in better understanding the processes of safeguarding websites and applications, and assist both students and faculty in learning about web security in a monitored school environment.

An application's flaws can be exploited or lead to an intrusion if it has defects or faults. Web apps are especially vulnerable to attack because of the immense worldwide advent of digital technology. These attacks can originate from various regions via a variety of offensive routes. A cybersecurity program would be incomplete without software security solutions and application vulnerability scans.

DVWA aims to teach students to identify the most prevalent web vulnerabilities, with difficulty levels ranging from easy to complex. You should be aware that this application has several known and unknown security flaws. You're expected to investigate as many problems as you can.

3.6 Atomic Red Team

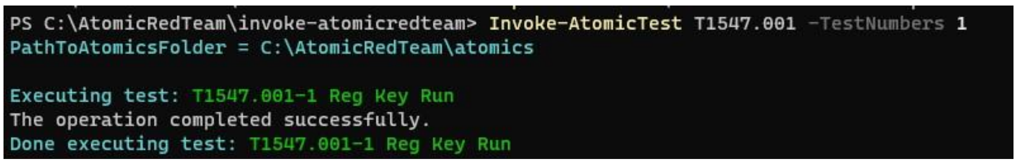
Security teams can use Atomic Red Team's open-source library of tests to replicate adversary behavior in their settings. There are few dependencies on the tests, and they are written in a way that can be automated. Atomic testing can be completed in five minutes with very little setup time. Spend more time testing rather than configuring. An approach based on "hopes and prayers" would not do for security teams who want to be proactive in their detection strategies. The MITRE ATT&CK matrix is mapped to atomic tests, so you always know what tactics you detect and what you don't. It is possible to mimic TTP in Detection Lab, see the telemetry generated, and make new detections using Atomic Red Team.

Atomic Red Team helps security teams analyze their environments or IDS/IPS technologies and form a gap analysis.

Atomic Red Team can be configured to use the execution framework (Invoke-Atomic Red Team) to conduct all or specific atomic tests mapped directly to the MITRE ATT&CK Framework in an automated fashion.

Invoke-Atomic Red Team is a PowerShell module to execute tests as defined in the atomics folder. The Invoke-Atomic Test function, followed by ATT&CK Technique and corresponding test numbers, can be used to run specific tests or a range of atomic tests. Passing the All option executes all atomic tests present within the atomics folder; however, this method is not recommended.

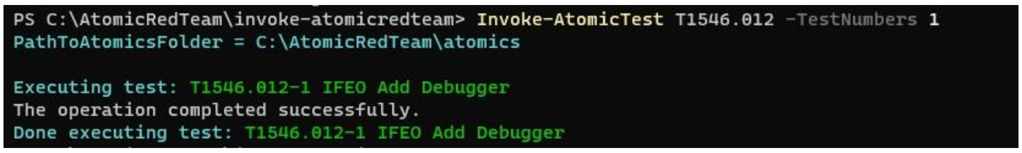
A usage example is shown below in Figure 15 where the Registry Run key ATT&CK technique was implemented using Atomic Red Team.



```
PS C:\AtomicRedTeam\invoke-atomicredteam> Invoke-AtomicTest T1547.001 -TestNumbers 1
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing test: T1547.001-1 Reg Key Run
The operation completed successfully.
Done executing test: T1547.001-1 Reg Key Run
```

Figure 15: *Executing ATT&CK T1547.001*

Similarly Figure 16 shows the execution of Image File Executions Options using Atomic Red Team, which is another ATT&CK Technique to establish Persistence in systems.



```
PS C:\AtomicRedTeam\invoke-atomicredteam> Invoke-AtomicTest T1546.012 -TestNumbers 1
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing test: T1546.012-1 IFEO Add Debugger
The operation completed successfully.
Done executing test: T1546.012-1 IFEO Add Debugger
```

Figure 16 *Executing ATT&CK T1546.012*

3.7 ESXi

VMware ESXi, a component of VMware's vSphere suite, is a bare-metal hypervisor that enables efficient virtualization. Unlike traditional operating systems, ESXi operates directly on server hardware without requiring an underlying OS, which reduces overhead and improves performance. ESXi manages multiple virtual machines (VMs), allowing each VM to run its own operating system and applications, thereby optimizing hardware utilization.

Key features of ESXi include:

1. **Resource Management:** ESXi provides advanced resource management capabilities, including CPU and memory allocation, to ensure optimal performance for VMs.
2. **Security:** It incorporates robust security features like Secure Boot, VM encryption, and role-based access control (RBAC) to protect the virtualized environment.
3. **High Availability and Fault Tolerance:** ESXi supports high availability (HA) and fault tolerance (FT), ensuring minimal downtime and continuous availability of critical applications.
4. **Scalability:** It can handle a large number of VMs and supports large-scale deployments, making it suitable for enterprise environments.
5. **Integration:** ESXi integrates seamlessly with VMware's ecosystem, including vCenter Server for centralized management, vSAN for storage, and NSX for network virtualization.

3.8 Terraform

Terraform is an open-source infrastructure as code (IaC) tool created by HashiCorp. It allows users to define and provision data center infrastructure using a high-level configuration language called HCL (HashiCorp Configuration Language).

Key features of Terraform include:

1. **Infrastructure as Code:** Terraform enables users to write code to manage infrastructure, making it easy to version, share, and manage configurations as illustrated in Figure 17.
2. **Provider Support:** It supports multiple cloud providers (e.g., AWS, Azure, Google Cloud), as well as on-premises solutions, through various providers.
3. **Plan and Apply:** Terraform generates an execution plan showing what it will do before making any changes. This helps in understanding the impact of changes.
4. **State Management:** It keeps track of the state of infrastructure, ensuring that the current state matches the desired configuration.
5. **Modularity:** Users can create reusable modules to organize infrastructure code and promote best practices.

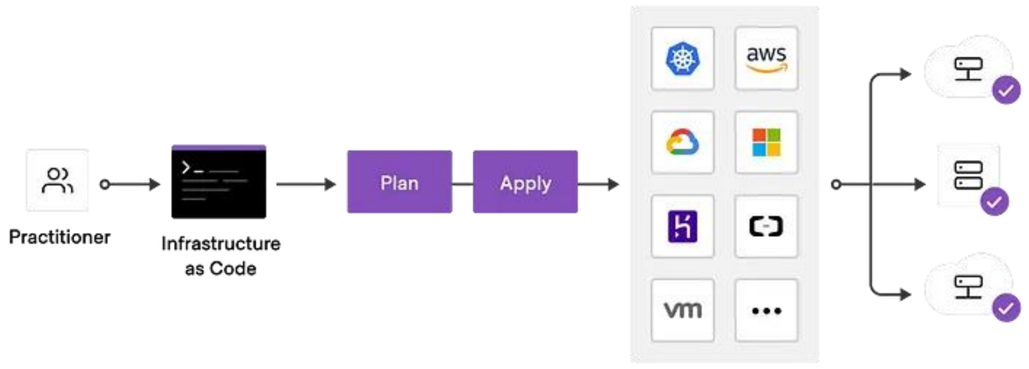


Figure 17: *Infrastructure of Terraform*

Chapter 4

Software Design Document (SDD)

4. Introduction

The Software Design Document (SDD) serves as a comprehensive guide outlining the architecture, functionality, and structure of a software system. It is a crucial artifact in the software development life cycle, providing a roadmap for developers, designers, and stakeholders to understand the details of the proposed system. By examining the design principles, modules, and interactions, the SDD aims to provide documentation aid in software development, detailing how the software should be built.

The purpose of the design document for "Hive Sentinel: The Hacker's Confine" entails a comprehensive overview covering technical and functional aspects to guide effective development. This chapter includes both textual and graphical details, such as the Block Flow diagram, Sequence diagrams, Database design, and Mock-ups of the web application. It outlines the high-level architecture of the system, defining main components, their relationships, and interactions, establishing the foundation for the entire project.

4.1 Project Modules

Project is divided into following modules for efficient development:

1. **Create Machines**
    - ➢ This module focuses on automating the process of creating machines through Terraform and deploying services on the created machines through Ansible from dashboard.

2. **Logs Generation and Storage**
    - ➢ This module focuses on generating logs on honeypots in response to event detected through Sysmon and Auditd.
    - ➢ Exporting logs on centralized database Elastic search.
    - ➢ Display logs on dashboard.

3. **MITRE ATT&CK Mapping**
    - ➢ This module focuses on mapping generated logs with MITRE ATT&CK Techniques.
    - ➢ Display MITRE ATT&CK mapping on dashboard.

4. **Identify Threat**
    - This module focuses on identifying threat from generated logs with the help of signature-based detection and displaying on dashboard.
    - Signature-based detection include comparing Sysmon logs with pre-defined attack list of python and identifying threats.

5. **Health Monitoring**
    - This module focuses on fetching health monitoring parameters from honeypots
    - Display health of honeypots on dashboard
    - Honeypot Re-spawning based on machine health

4.2 Block Flow Diagram

A block flow diagram for the project, "Hive Sentinel: The Hacker's Confines," offers a simplified representation of the overall structure and key components illustrated in Figure 18. Labelled blocks signify crucial stages, while arrows illustrate the flow or sequence of actions. This high-level overview aids in visualizing and comprehending the project's structure and functionality.

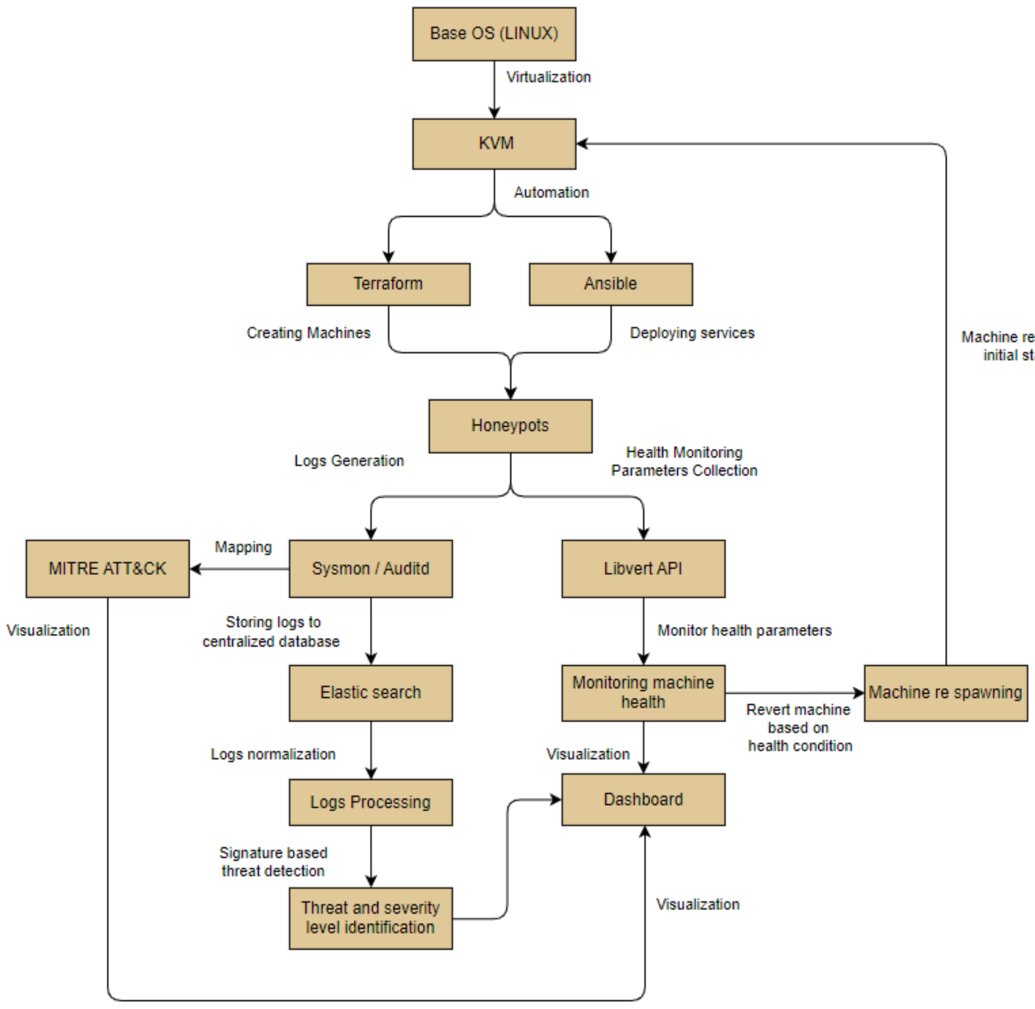


***Figure 18**: Block Flow Diagram*

As depicted in Figure 18, the initial phase involves establishing the foundational environment by installing the Linux operating system and implementing KVM for virtualization. The infrastructure deployment follows a systematic approach using Terraform and Ansible. Terraform is employed to define the infrastructure as code and deploy virtual machines and services on KVM, while Ansible is used for deploying services on virtual machines.

Moving on to the log generation path, Sysmon/Auditd configurations are implemented on the virtual machines to generate logs, which are then mapped to MITRE ATT&CK for visualization. The logs are pushed to Elasticsearch for storage, normalized for consistency, processed to identify relevant information, and subjected to signature-based threat detection with severity levels identified. The resulting data is displayed on a dashboard that includes MITRE ATT&CK mapping, threat information, and severity levels.

In parallel, the health monitoring path utilizes Libvirt API for fetching health monitoring parameters from machines, displaying them on the dashboard. Thresholds are set for health parameters, and if breached, a snapshot is applied, and machines are respawned within the KVM environment. The KVM state is then updated to reflect the changes. This comprehensive approach ensures a robust and secure virtualized environment with effective log generation, threat detection, and health monitoring.

4.3 Sequence Diagrams

Sequence diagrams for the project, "Hive Sentinel: The Hacker's Confines," illustrates the interactions and chronological order of messages exchanged among objects or components within a system. They provide a dynamic view of how various entities collaborate to accomplish a specific task or process.

4.3.1 Login

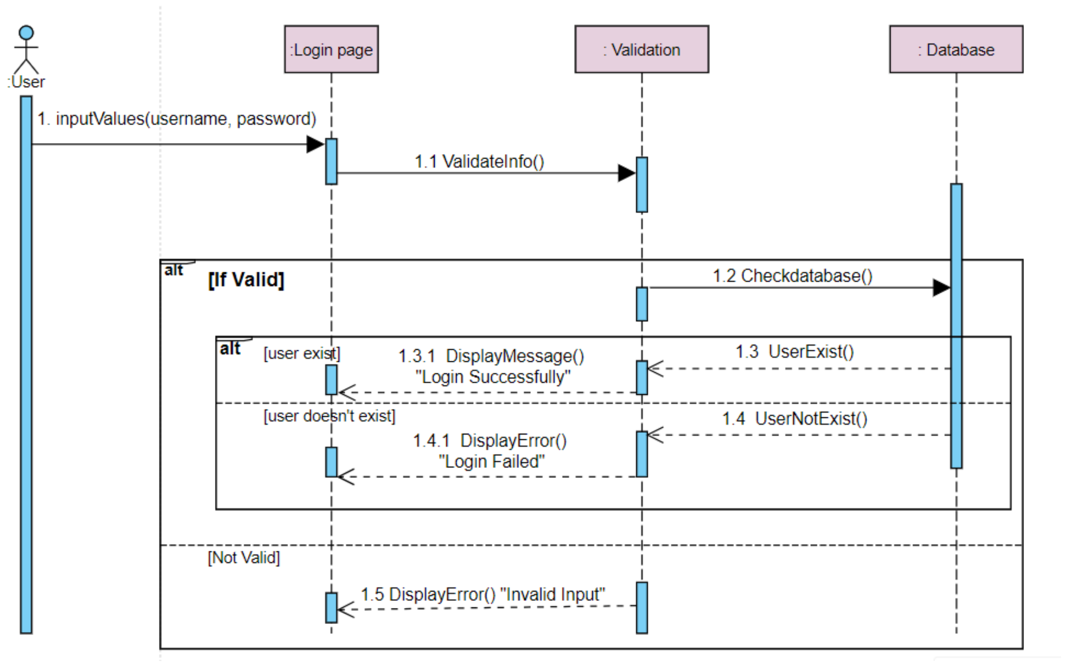


Figure 19: *Sequence diagram of login page*

Sequence of actions as illustrated in Figure 19 are described below:

1. User will enter the required fields username and password in Login page. After clicking Login button, the Login page will check the validations.
2. If the user doesn't fill all the required data in form or submit wrong information that is against the validation rules, error message will be displayed "invalid input" and system will remain on login page.
3. After correctly entering the Login credentials, it will be checked in database for authentication.
4. If the user exist in database following message will be displayed on screen "Login Successfully" and will move to Overview page.
5. If the user doesn't exist in database following message will be displayed "Login Failed".

4.3.2 Create Machines

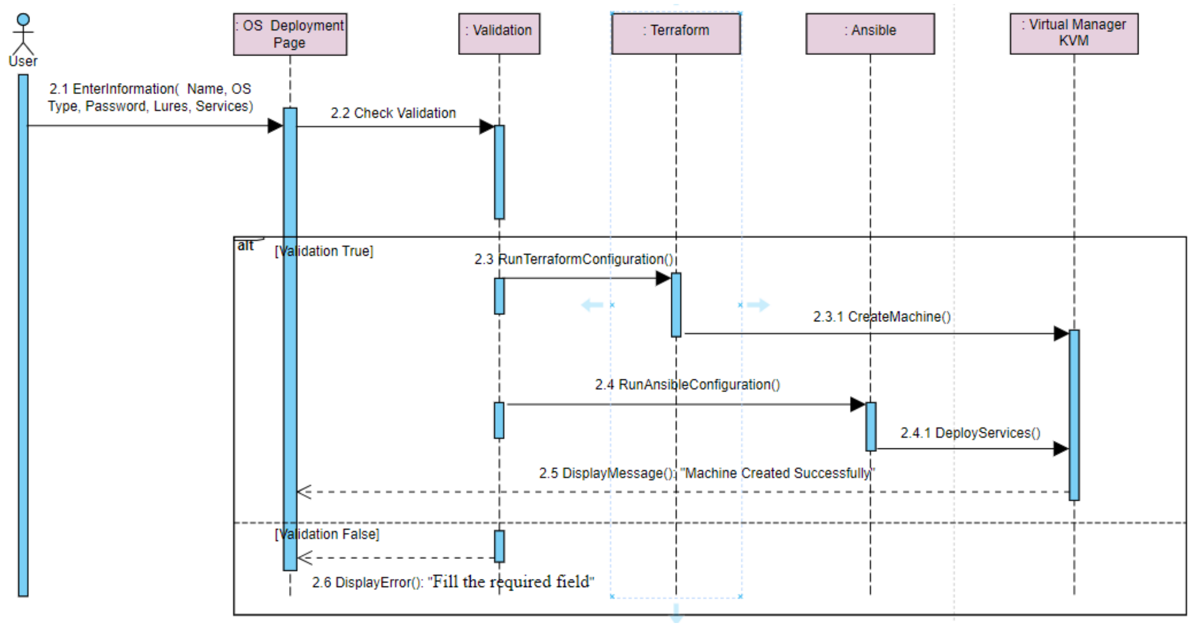

User
: OS Deployment Page
: Validation
: Terraform
: Ansible
: Virtual Manager KVM
2.1 EnterInformation(Name, OS Type, Password, Lures, Services)
2.2 Check Validation
alt
[Validation True]
2.3 RunTerraformConfiguration()
2.3.1 CreateMachine()
2.4 RunAnsibleConfiguration()
2.4.1 DeployServices()
2.5 DisplayMessage(): "Machine Created Successfully"
[Validation False]
2.6 DisplayError(): "Fill the required field"

***Figure 20**: Sequence diagram of OS Deployment page*

Sequence of actions as illustrated in Figure 20 are described below:

1. User will enter the required fields Name of machine, OS type, Password, Services and Lures in OS Deployment page. After clicking create button, the OS Deployment page will check the validations.
2. If the user doesn't fill all the required data in form, system will display error message "Fill the required field" and will remain on OS deployment page.
3. After correctly entering all the required fields, Terraform configuration will run to create machine on KVM.
4. After successfully machine is created, Ansible configuration will run to deploy services on machine.

4.3.3 View Deployed Machines

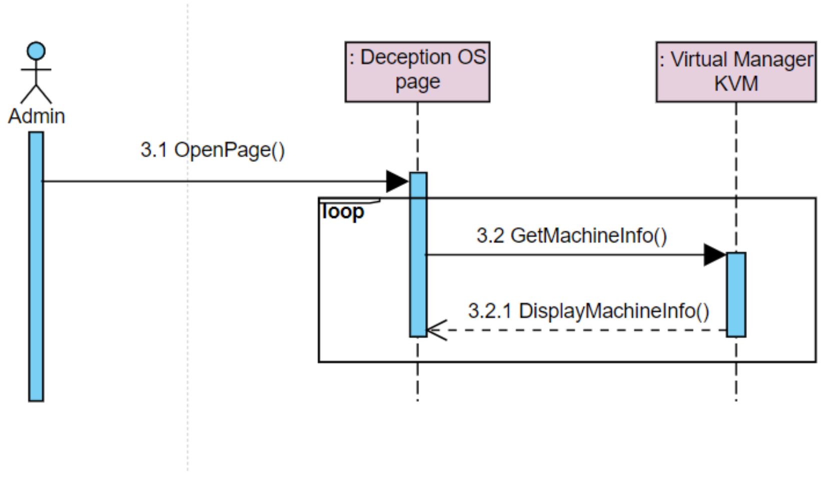


Figure 21: *Sequence diagram of Deception OS page*

Sequence of actions as illustrated in Figure 21 are described below:

1. User will open Deception OS page.

2. All the information regarding deployed machines will be fetched from Virtual Manager and displayed on page.

4.3.4 Logs Analysis

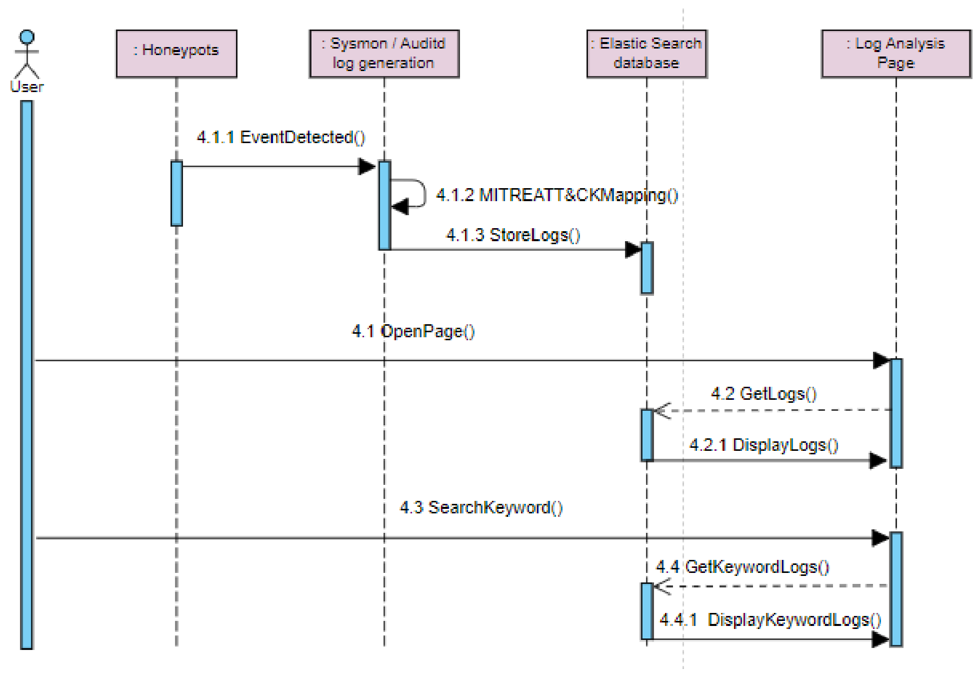


Figure 22: *Sequence diagram of Log Analysis page*

Sequence of actions as illustrated in Figure 22 are described below:

1. User will open Log Analysis page.

2. Logs are generated by Sysmon or Auditd on the honeypot with MITRE ATT&CK mapping, and then these logs are exported or pushed to the Elastic search database. Subsequently, they can be fetched from the Elastic search database and displayed on the dashboard.

4.3.5 MITRE ATT&CK Mapping

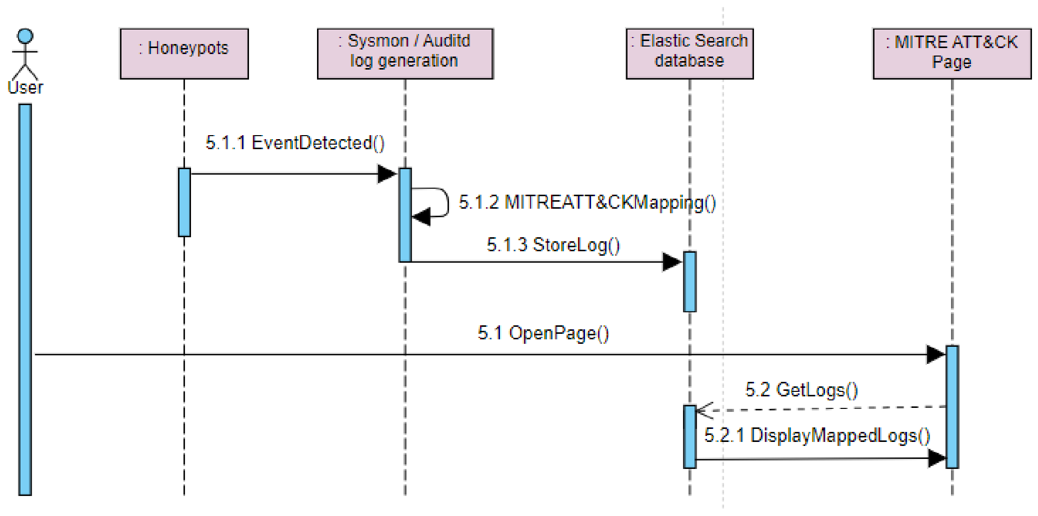

User
: Honeypots
: Sysmon / Auditd log generation
: Elastic Search database
: MITRE ATT&CK Page
5.1.1 EventDetected()
5.1.2 MITREATT&CKMapping()
5.1.3 StoreLog()
5.1 OpenPage()
5.2 GetLogs()
5.2.1 DisplayMappedLogs()

Figure 23: *Sequence diagram of MITRE ATT&CK page*

Sequence of actions as illustrated in Figure 23 are described below:

1. User will open MITTRE ATT&CK page.

2. Generated logs with MITRE ATT&CK mapping will be fetched from Elastic search database and will be displayed on dashboard.

4.3.6 Health Monitoring

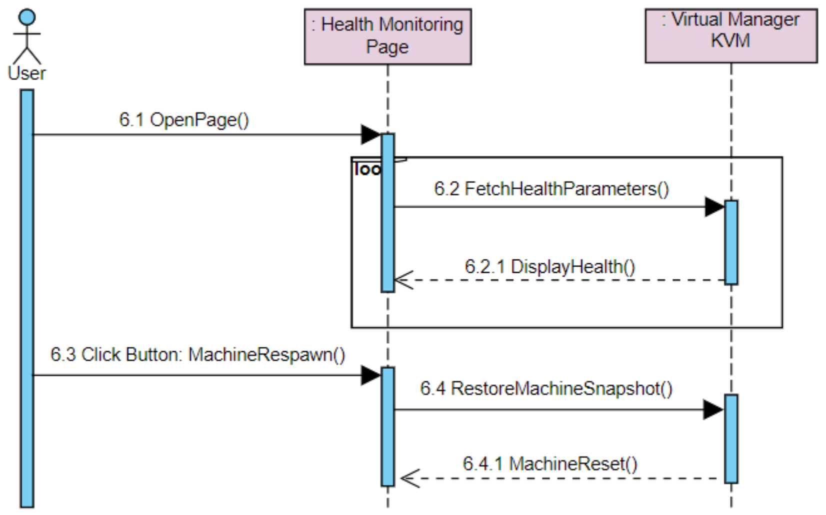


Figure 24: *Sequence diagram of Health Monitoring page*

Sequence of actions as illustrated in Figure 24 are described below:

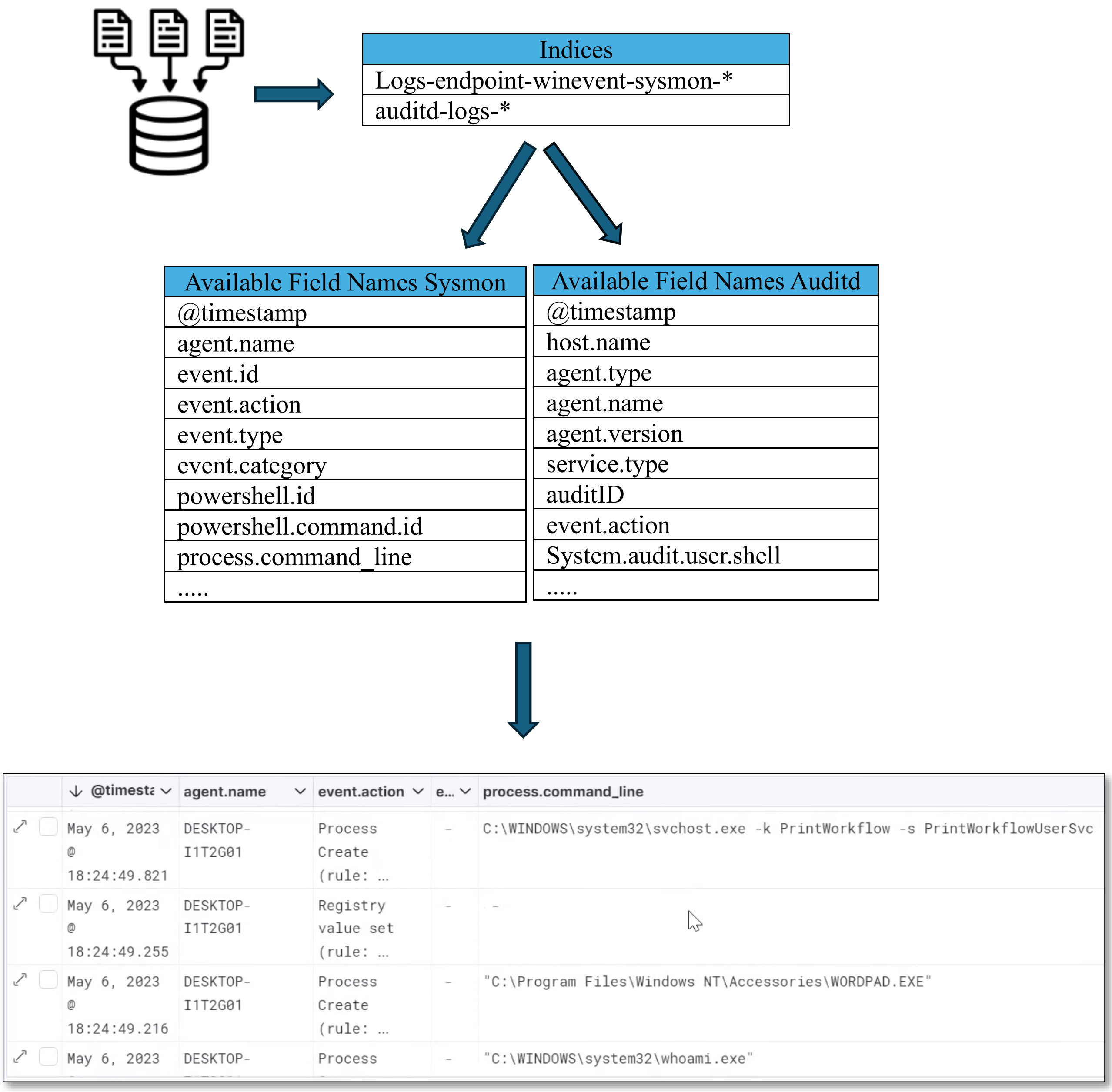
1. User will open Health Monitoring page.

2. Health Monitoring parameters such as CPU and memory usage of each machine in Virtual Manager will be fetched through Libvirt API and displayed on dashboard.

3. Thresholds are set for health parameters, and if breached, a snapshot is applied, and machines are respawned within the KVM environment to its initial state.

4.4 Database Design

Hive Sentinel incorporates Elasticsearch database for its powerful search and analytics capabilities, making it an ideal choice for handling vast datasets. Its efficient storage and retrieval mechanisms, based on distributed indexing and sharding, enable quick and scalable data access. Elasticsearch excels at handling both structured and unstructured data in real-time, making it valuable for applications such as log analysis. Its ability to deliver fast and relevant search results, coupled with distributed architecture for resilience, makes Elasticsearch a popular choice for organizations seeking robust data exploration and retrieval solutions. Elasticsearch will create different indices for incoming log files from Sysmon and Auditd from various honeypots. A list of available field names processed by Elasticsearch will be provided. We can then create a table view using the suggested field names based on our specific requirements.



Indices
Logs-endpoint-winevent-sysmon-*
auditd-logs-*

Available Field Names Sysmon
@timestamp
agent.name
event.id
event.action
event.type
event.category
powershell.id
powershell.command.id
process.command_line
.....

Available Field Names Auditd
@timestamp
host.name
agent.type
agent.name
agent.version
service.type
auditID
event.action
System.audit.user.shell
.....

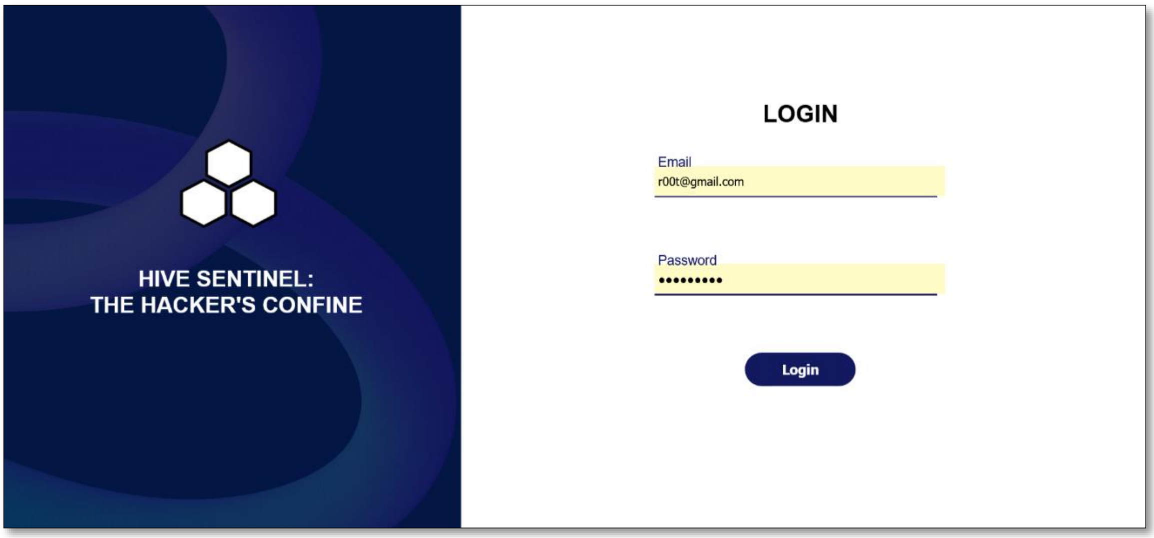
@timesta	agent.name	event.action	e...	process.command_line
May 6, 2023 @ 18:24:49.821	DESKTOP-I1T2G01	Process Create (rule: ...	-	C:\WINDOWS\system32\svchost.exe -k PrintWorkflow -s PrintWorkflowUserSvc
May 6, 2023 @ 18:24:49.255	DESKTOP-I1T2G01	Registry value set (rule: ...	-	-
May 6, 2023 @ 18:24:49.216	DESKTOP-I1T2G01	Process Create (rule: ...	-	"C:\Program Files\Windows NT\Accessories\WORDPAD.EXE"
May 6, 2023	DESKTOP-	Process	-	"C:\WINDOWS\system32\whoami.exe"

Figure 25: *Database design*

4.5 System User Interface

4.5.1 Login Page

The first page the user interacts with is the Login page, serving as the gateway to access the software's features and functionalities. Here, users input their username and password. Upon clicking the Login button, the system conducts validation checks, ensuring accurate data entry. If any errors occur, such as incomplete fields or incorrect information, an error message is promptly displayed, guiding users to rectify their input. Upon successful entry, the system authenticates the user's credentials against the database. If a match is found, a reassuring "Login Successfully" message appears, granting access to the Overview page. However, if the user's credentials do not match records in the database, a "Login Failed" message is displayed, signaling the need for further action. Figure 26 illustrates the user interface of login page.



Email: r00t@gmail.com

Password: •••••••••

Figure 26: *Login Page*

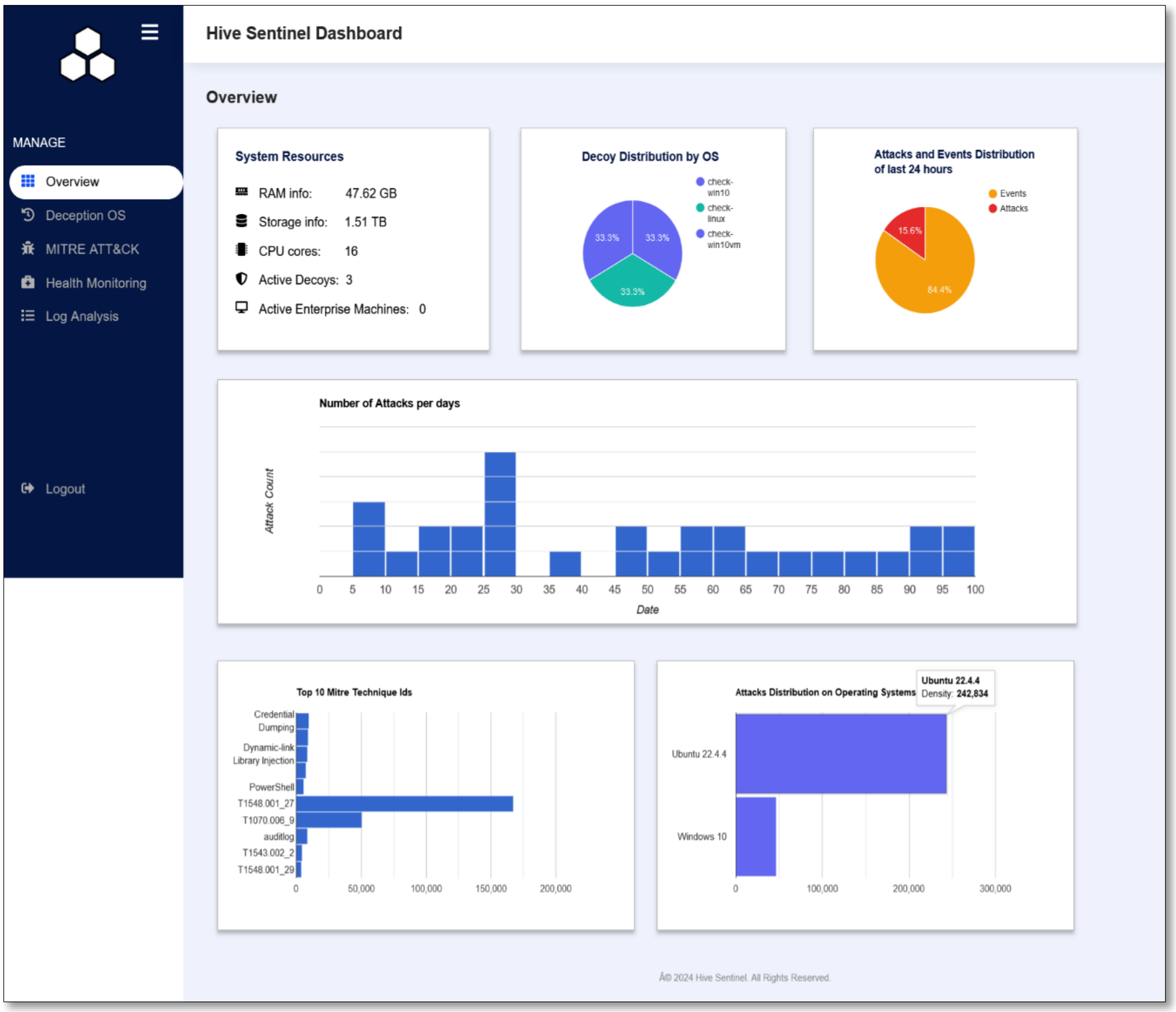
4.5.2 Overview Page

After Successful login user will be redirected to overview page that contains the analysis and statistics of machines, events and attacks. Following are the charts displayed on overview page:

1. System Resources that contain the information of total RAM of server, total hard disk storage, CPU cores, active decoys and active enterprise machines
2. Decoy Distribution by Operating System that gives information in the form of pie chart about total number of deployed honeypots and their operating system.

3. Attacks and Events distribution by all OS from last 24 hours
4. Number of attacks per days from last one month
5. Top 10 MITRE technique ids from deployed honeypots of Linux and windows.
6. Attacks distribution on honeypots of Linux and windows.

Figure 27 illustrates the user interface of the overview page (dashboard).


Hive Sentinel Dashboard

Overview

System Resources

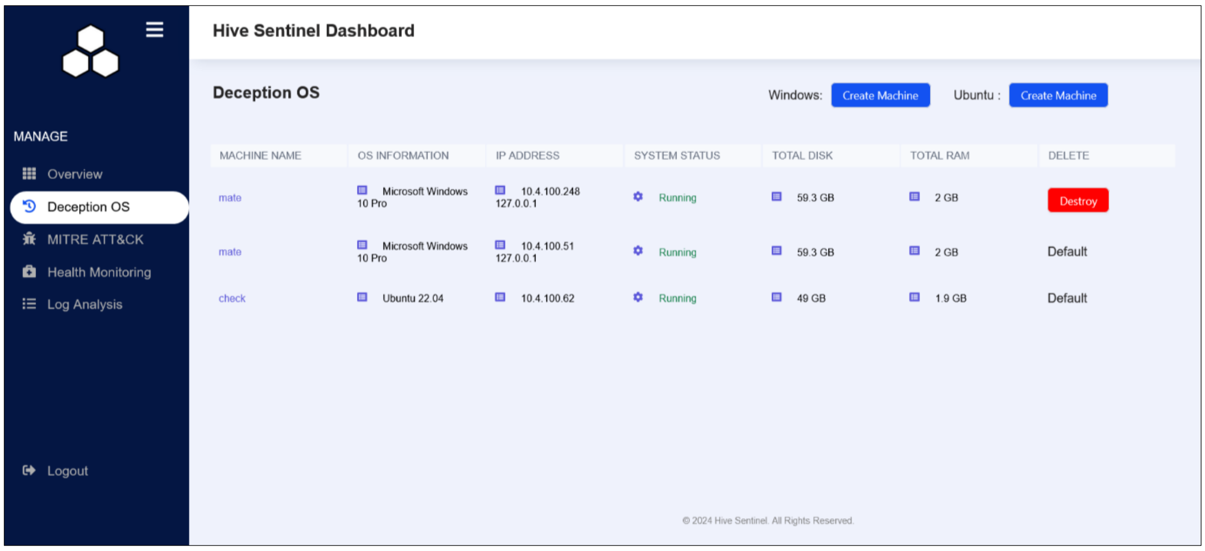
RAM info:	47.62 GB
Storage info:	1.51 TB
CPU cores:	16
Active Decoys:	3
Active Enterprise Machines:	0

Figure 27: Overview Page

4.5.3 Deception OS Page

Deception OS page has two main functionalities:

1. It displays the information like hostname, ip address, OS information, system status, RAM and hardisk of deployed honeypots
2. It provides the functionality of creating honeypots of windows and linux with create machine button. Deployed machine can be successfully deleted by clicking on destroy button. By clicking the create and destroy button, terraform script will run that will automatically create or destroy honeypot on server. Figure 28 illustrates the user interface of deception OS page.



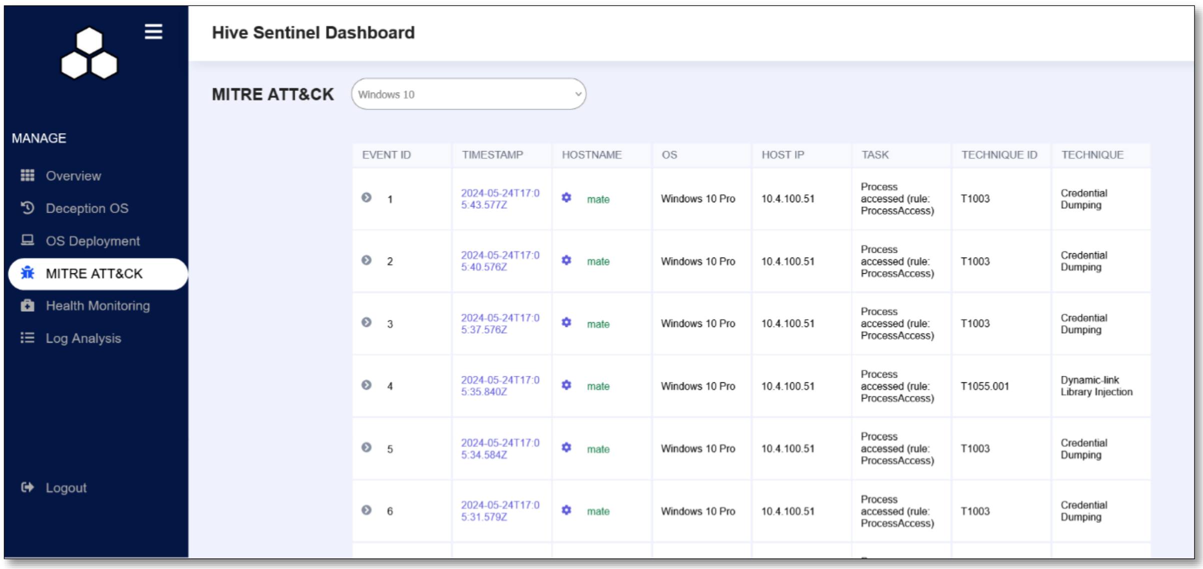
Deception OS

MACHINE NAME	OS INFORMATION	IP ADDRESS	SYSTEM STATUS	TOTAL DISK	TOTAL RAM	DELETE
mate	Microsoft Windows 10 Pro	10.4.100.248 127.0.0.1	Running	59.3 GB	2 GB	Destroy
mate	Microsoft Windows 10 Pro	10.4.100.51 127.0.0.1	Running	59.3 GB	2 GB	Default
check	Ubuntu 22.04	10.4.100.62	Running	49 GB	1.9 GB	Default

Figure 28: *Deception OS Page*

4.5.4 MITRE ATT&CK Page

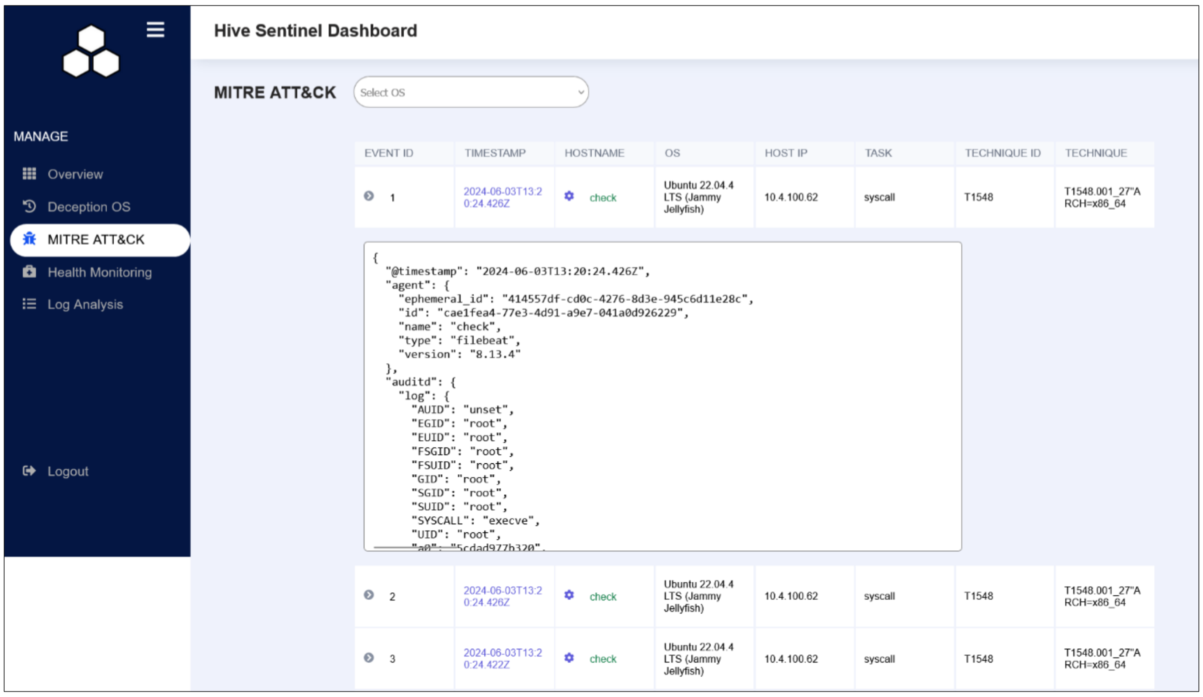
By opening this page, generated logs with MITRE ATT&CK mapping will be fetched from Elastic search database and will be displayed on dashboard. Figure 29 illustrates the user interface of MITRE ATT&CK page.



MITRE ATT&CK

Windows 10

EVENT ID	TIMESTAMP	HOSTNAME	OS	HOST IP	TASK	TECHNIQUE ID	TECHNIQUE
1	2024-05-24T17:05:43.577Z	mate	Windows 10 Pro	10.4.100.51	Process accessed (rule: ProcessAccess)	T1003	Credential Dumping
2	2024-05-24T17:05:40.576Z	mate	Windows 10 Pro	10.4.100.51	Process accessed (rule: ProcessAccess)	T1003	Credential Dumping
3	2024-05-24T17:05:37.576Z	mate	Windows 10 Pro	10.4.100.51	Process accessed (rule: ProcessAccess)	T1003	Credential Dumping
4	2024-05-24T17:05:35.840Z	mate	Windows 10 Pro	10.4.100.51	Process accessed (rule: ProcessAccess)	T1055.001	Dynamic-link Library Injection
5	2024-05-24T17:05:34.584Z	mate	Windows 10 Pro	10.4.100.51	Process accessed (rule: ProcessAccess)	T1003	Credential Dumping
6	2024-05-24T17:05:31.579Z	mate	Windows 10 Pro	10.4.100.51	Process accessed (rule: ProcessAccess)	T1003	Credential Dumping



MITRE ATT&CK

Select OS

EVENT ID	TIMESTAMP	HOSTNAME	OS	HOST IP	TASK	TECHNIQUE ID	TECHNIQUE
1	2024-06-03T13:20:24.426Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	syscall	T1548	T1548.001_27"ARCH=x86_64

```
{
  "@timestamp": "2024-06-03T13:20:24.426Z",
  "agent": {
    "ephemeral_id": "414557df-cd0c-4276-8d3e-945c6d11e28c",
    "id": "cae1fea4-77e3-4d91-a9e7-041a0d926229",
    "name": "check",
    "type": "filebeat",
    "version": "8.13.4"
  },
  "auditd": {
    "log": {
      "AUID": "unset",
      "EGID": "root",
      "EUID": "root",
      "FSGID": "root",
      "FSUID": "root",
      "GID": "root",
      "SGID": "root",
      "SUID": "root",
      "SYSCALL": "execve",
      "UID": "root",
      "a0": "5cdad977b320",
```

EVENT ID	TIMESTAMP	HOSTNAME	OS	HOST IP	TASK	TECHNIQUE ID	TECHNIQUE
2	2024-06-03T13:20:24.426Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	syscall	T1548	T1548.001_27"ARCH=x86_64
3	2024-06-03T13:20:24.422Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	syscall	T1548	T1548.001_27"ARCH=x86_64

***Figure 29:** MITRE ATT&CK Page*

4.5.5 Health Monitoring Page

The Health Monitoring page provides a feature to keep track of important system metrics. These metrics of every machine are linked with elastic search database and status of these health metrics are updated in database after every 20 seconds. Health parameters such as CPU usage, memory usage, disk usage, load gauge 5 minutes, packet loss, inbound traffic, and outbound traffic are monitored. Thresholds are set for these parameters, and if any threshold is exceeded, a snapshot is taken, and the affected machines are restored to their original state within the ESXI environment. This process ensures that the system remains stable and responsive, maintaining optimal performance and minimizing disruptions. Figure 30 illustrates the user interface of health monitoring page.

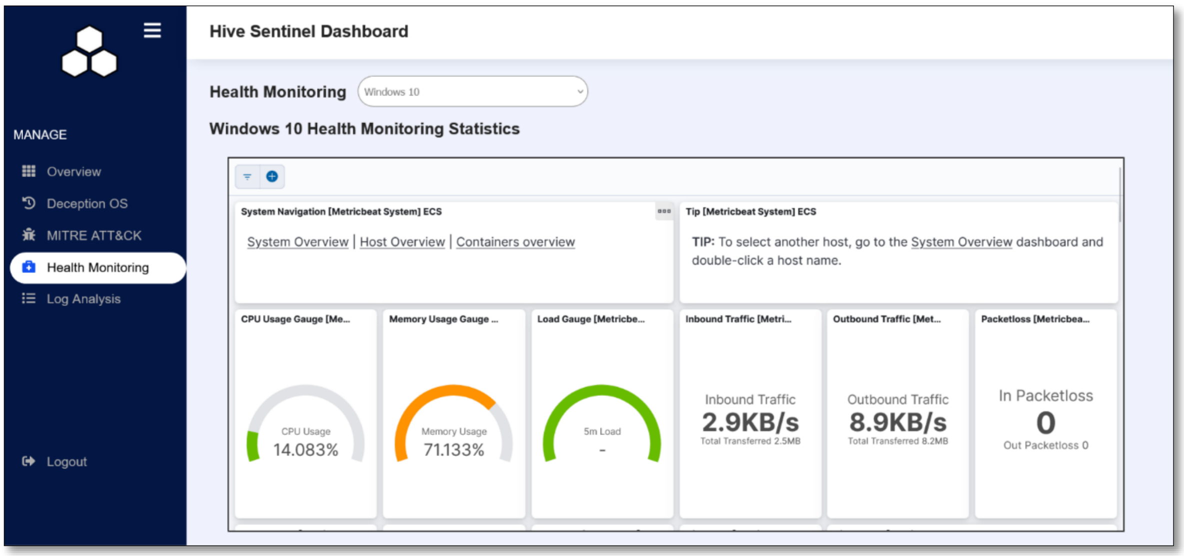
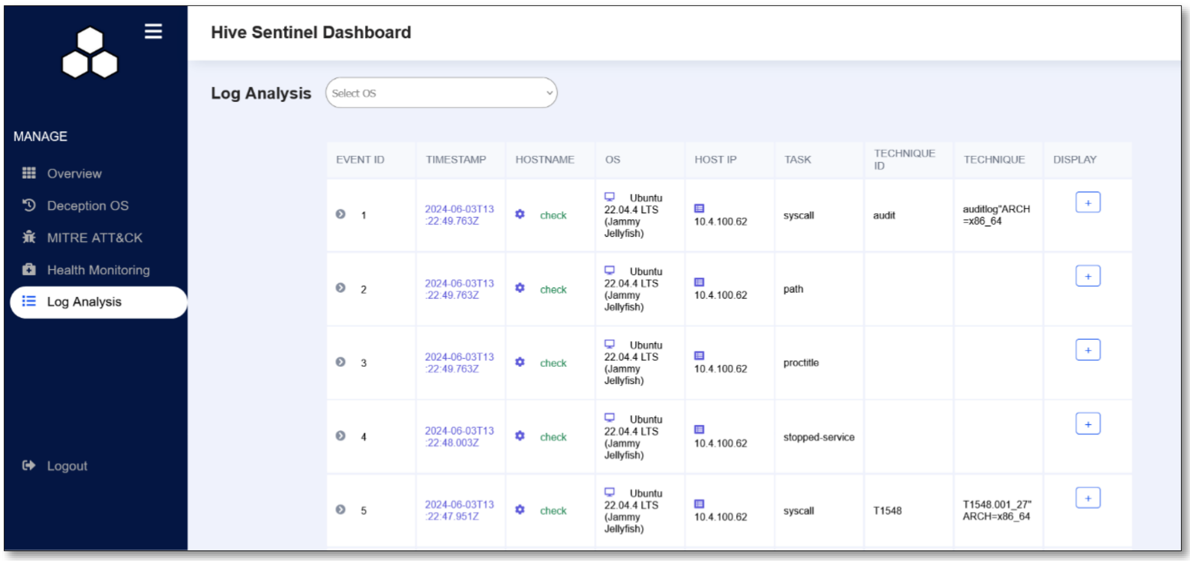


Figure 30: *Health Monitoring Page*

4.5.6 Log Analysis Page

By opening this page, all generated logs including MITRE ATT&CK mapping logs will be fetched from Elastic search database and will be displayed on dashboard. Figure 31 illustrates the user interface of log analysis page.

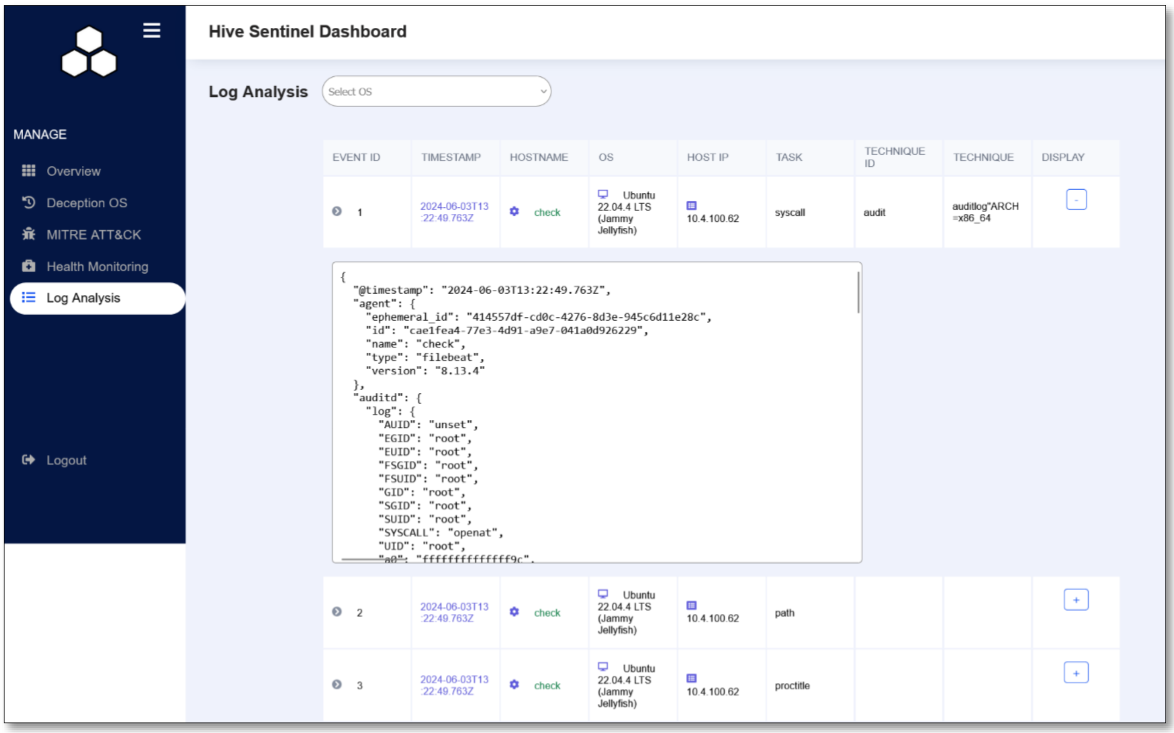


Hive Sentinel Dashboard

Log Analysis

Select OS

EVENT ID	TIMESTAMP	HOSTNAME	OS	HOST IP	TASK	TECHNIQUE ID	TECHNIQUE	DISPLAY
1	2024-06-03T13:22:49.763Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	syscall	audit	auditlog"ARCH=x86_64	+
2	2024-06-03T13:22:49.763Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	path			+
3	2024-06-03T13:22:49.763Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	proctitle			+
4	2024-06-03T13:22:48.003Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	stopped-service			+
5	2024-06-03T13:22:47.951Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	syscall	T1548	T1548.001_27"ARCH=x86_64	+



Hive Sentinel Dashboard

Log Analysis

Select OS

EVENT ID	TIMESTAMP	HOSTNAME	OS	HOST IP	TASK	TECHNIQUE ID	TECHNIQUE	DISPLAY
1	2024-06-03T13:22:49.763Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	syscall	audit	auditlog"ARCH=x86_64	-

```
{
  "@timestamp": "2024-06-03T13:22:49.763Z",
  "agent": {
    "ephemeral_id": "414557df-cd0c-4276-8d3e-945c6d11e28c",
    "id": "cae1fea4-77e3-4d91-a9e7-041a0d926229",
    "name": "check",
    "type": "filebeat",
    "version": "8.13.4"
  },
  "auditd": {
    "log": {
      "AUID": "unset",
      "EGID": "root",
      "EUID": "root",
      "FSGID": "root",
      "FSUID": "root",
      "GID": "root",
      "SGID": "root",
      "SUID": "root",
      "SYSCALL": "openat",
      "UID": "root",
      "a0": "ffffffffffffff9c",
```

EVENT ID	TIMESTAMP	HOSTNAME	OS	HOST IP	TASK	TECHNIQUE ID	TECHNIQUE	DISPLAY
2	2024-06-03T13:22:49.763Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	path			+
3	2024-06-03T13:22:49.763Z	check	Ubuntu 22.04.4 LTS (Jammy Jellyfish)	10.4.100.62	proctitle			+

Figure 31: *Log Analysis Page*

Chapter 5

Project Methodology and Implementation

5. Introduction

In this chapter, we explore the methodology and implementation of the Hive Sentinel project. By leveraging virtualization with ESXi and automation through Terraform, we streamline the deployment of honeypots. The integration of Sysmon and Auditd with the MITRE ATT&CK framework enhances security monitoring. Logs are centralized in Elasticsearch, enabling efficient storage and analysis. A dynamic React.js dashboard, supported by a Python and Flask backend, provides real-time visualization and management of honeypots. This comprehensive approach ensures a robust and scalable honeypot system for detecting and analyzing cyber threats.

5.1 Project Methodology

To efficiently execute the process of creating Hive Sentinel, a robust combination of frameworks and tools is essential. Figure 6.1 presents the actions and activities that are necessary for achieving an effective deployment of a honeypot system. A sequence of activities is illustrated in Figure 6.1 and discussed below.

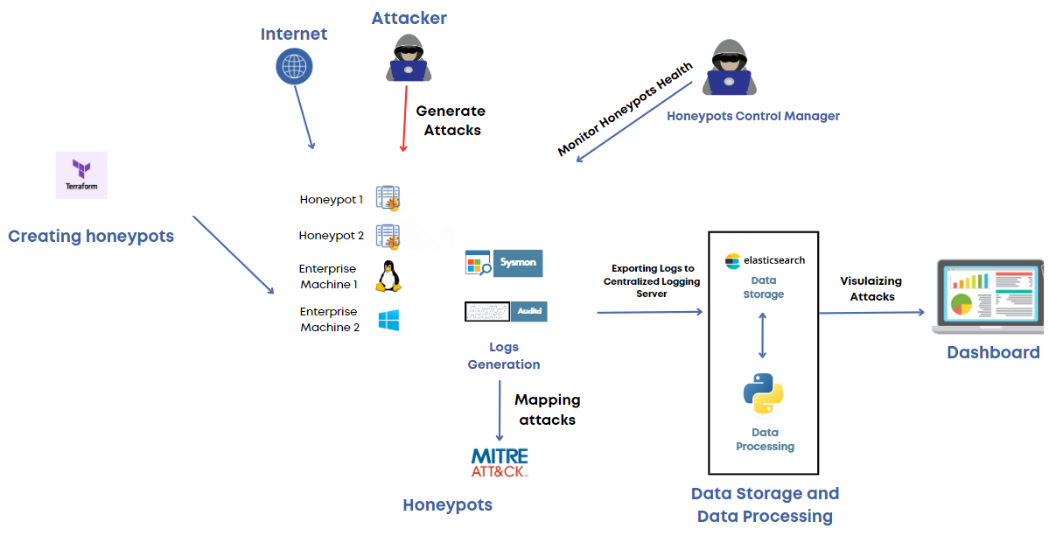


Figure 32: *Project Architecture Diagram*

As illustrated in Figure 32, the project begins with virtualization; ESXi facilitates hardware virtualization, allowing the concurrent operation of multiple virtual machines on a single host. For automation, Terraform streamlines infrastructure provisioning and management through code, helping to automatically create virtual machines on ESXi.

Process monitoring involves implementing Sysmon on Windows and Auditd on Linux, enhancing system security through detailed logging of process executions and network connections.

Security infrastructure integration is achieved through the incorporation of MITRE ATT&CK with generated logs from Sysmon and Auditd, providing a comprehensive framework for understanding cyber adversary tactics and techniques. After successfully generating logs and mapping them with the MITRE ATT&CK framework, export the logs to a centralized Elasticsearch database. Elasticsearch efficiently stores and indexes large datasets, enabling centralized monitoring.

Process the logs stored in Elasticsearch through Python to ensure they are in the same format, as they come from different honeypot machines running Windows and Linux. Visualization is facilitated by developing a dynamic React.js dashboard with a robust Python and Flask backend to monitor health parameters of honeypots, create machines, analyze MITTRE ATT&CK techniques, and control management of honeypots.

This integrated approach ensures a comprehensive cybersecurity framework, seamlessly combining virtualization, automation, monitoring, database management, security integration, and visualization, thereby ensuring an efficient honeypot system.

5.2 Project Implementation

As discussed in project methodology, Hive Sentinel: The Hacker's Confine development is divided into following modules:

1. Virtualization

2. Automation through Terraform

3. Event Management Tool Configuration with MITRE ATT&CK Framework

5. Elasticsearch Integration

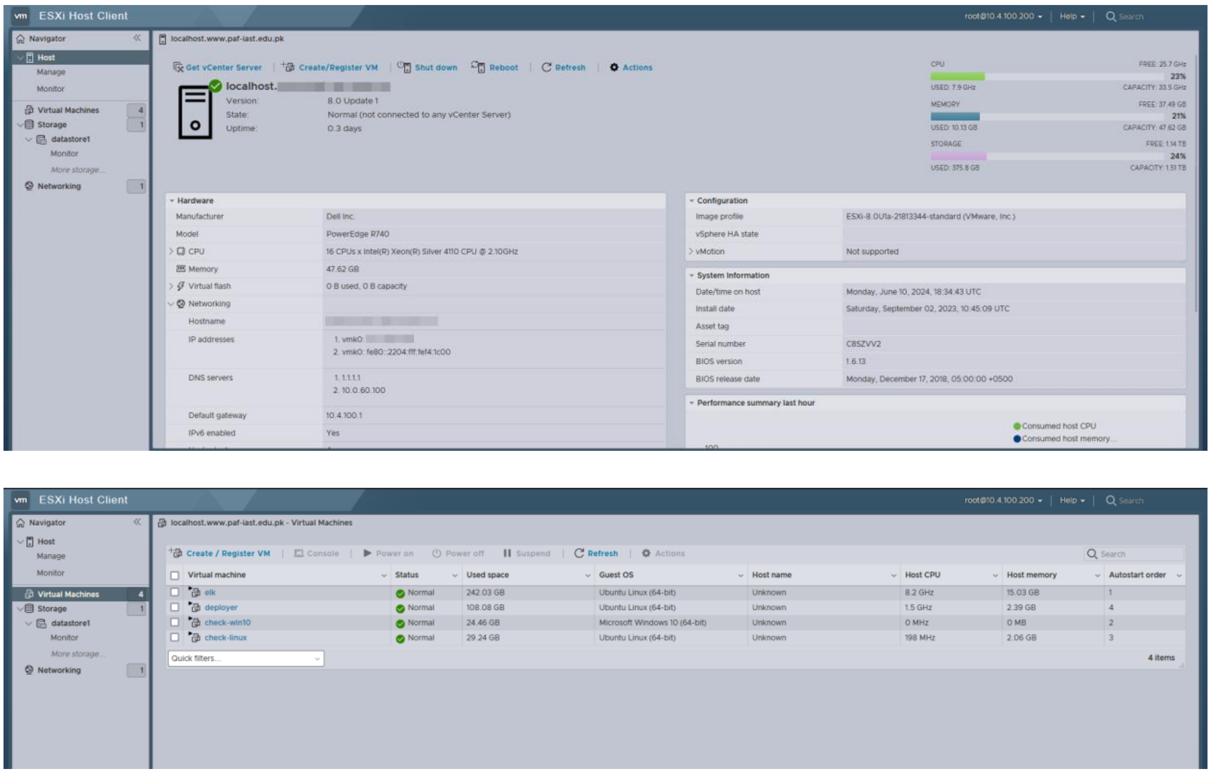
5.2.1 Virtualization

Virtualization is the first step in Hive Sentinel development, where honeypots are deployed to simulate vulnerable environments for detecting and analyzing malicious activities. By creating isolated and controlled virtual environments, we can effectively study and respond to cyber threats without compromising actual systems.

5.2.1.1 ESXi

ESXi is a type-1 hypervisor developed by VMware that provides robust and efficient virtualization capabilities. It allows multiple virtual machines to run concurrently on a single physical server, optimizing resource utilization and providing a scalable solution for deploying honeypots. Leveraging ESXi's advanced features ensures that the virtualization layer of Hive Sentinel is both reliable and efficient, forming a strong foundation for the subsequent modules of the project.

Figure 33 illustrates the deployment of virtual machines on ESXI



localhost.www.paf-iast.edu.pk

localhost.[redacted]

Version:	8.0 Update 1
State:	Normal (not connected to any vCenter Server)
Uptime:	0.3 days

CPU	USED: 7.9 GHz	FREE: 25.7 GHz 23% CAPACITY: 33.5 GHz
MEMORY	USED: 10.13 GB	FREE: 37.49 GB 21% CAPACITY: 47.62 GB
STORAGE	USED: 375.8 GB	FREE: 1.14 TB 24% CAPACITY: 1.51 TB

Hardware	
Manufacturer	Dell Inc.
Model	PowerEdge R740
CPU	16 CPUs x Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz
Memory	47.62 GB
Virtual flash	0 B used, 0 B capacity
Networking	
Hostname	[redacted]
IP addresses	1. vmk0: [redacted] 2. vmk0: fe80::2204:fff:fef4:1c00
DNS servers	1. 1.1.1.1 2. 10.0.60.100
Default gateway	10.4.100.1
IPv6 enabled	Yes

Configuration	
Image profile	ESXi-8.0U1a-21813344-standard (VMware, Inc.)
vSphere HA state	
vMotion	Not supported

System Information	
Date/time on host	Monday, June 10, 2024, 18:34:43 UTC
Install date	Saturday, September 02, 2023, 10:45:09 UTC
Asset tag	
Serial number	C8SZVV2
BIOS version	1.6.13
BIOS release date	Monday, December 17, 2018, 05:00:00 +0500

Performance summary last hour

localhost.www.paf-iast.edu.pk - Virtual Machines

Virtual machine	Status	Used space	Guest OS	Host name	Host CPU	Host memory	Autostart order
elk	Normal	242.03 GB	Ubuntu Linux (64-bit)	Unknown	8.2 GHz	15.03 GB	1
deployer	Normal	108.08 GB	Ubuntu Linux (64-bit)	Unknown	1.5 GHz	2.39 GB	4
check-win10	Normal	24.46 GB	Microsoft Windows 10 (64-bit)	Unknown	0 MHz	0 MB	2
check-linux	Normal	29.24 GB	Ubuntu Linux (64-bit)	Unknown	198 MHz	2.06 GB	3

4 items

***Figure 33:** Deployment of virtual machines on ESXI*

Description of deployed machines on ESXI are as follow:

1. ELK (Elastic Search + Kibana)

The ELK machine on ESXi hosts Elastic Search and Kibana. Elastic Search is a distributed, scalable search engine for real-time data indexing and querying.

Kibana, running on the same VM, provides a web-based interface for visualizing and analyzing the data stored in Elastic Search, allowing users to create dynamic dashboards and reports.

2. Deployer (Terraform + Dashboard)

The Deployer machine on ESXi integrates Terraform and a customizable dashboard for managing infrastructure as code. Terraform automates the provisioning and management of cloud resources across various providers. The dashboard provides a centralized view for monitoring deployments, managing resources, and visualizing infrastructure changes, enhancing operational efficiency from a single VM.

3. Check-Linux (Linux Honeypot)

The Check-Linux machine on ESXi is a security-focused Linux honeypot designed to detect, monitor, and analyze malicious activities. It simulates a vulnerable Linux environment to attract attackers, logging their actions to provide valuable insights into attack techniques and patterns. This helps in improving security measures and understanding emerging threats, all within a controlled VM environment.

4. Check-Win10 (Windows Honeypot)

The Check-Win10 machine on ESXi is a Windows honeypot configured to mimic a standard Windows 10 system. It lures cyber attackers, capturing their behavior and tactics. By recording and analyzing these interactions, it aids in identifying vulnerabilities, understanding attack vectors, and enhancing the overall security posture against Windows-targeted threats within a VM setup.

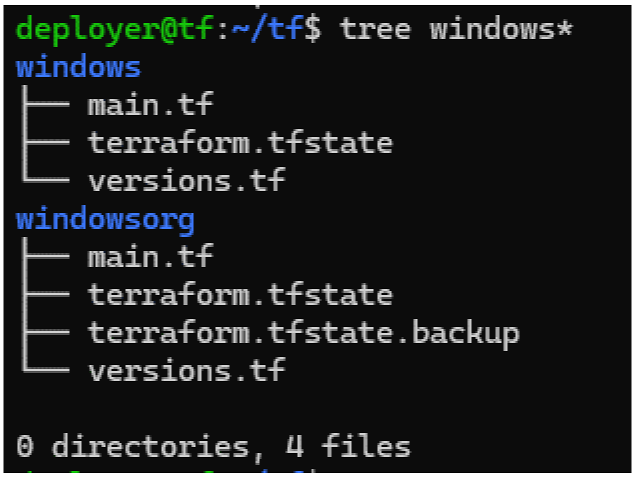
5.2.2 Automation through Terraform

Terraform is utilized to streamline the provisioning and management of the infrastructure within Hive Sentinel. By using infrastructure as code (IaC), Terraform allows for the automated and consistent creation, configuration, and deployment of virtual machines on ESXi. Configuration files define the desired state of the infrastructure, which Terraform then applies, ensuring that the environment is reproducible and scalable. This automation reduces manual intervention, minimizes errors, and enhances the efficiency of managing complex virtual environments for deploying honeypots and other components of the project.

5.2.2.1 Windows Machine Terraform Configuration:

```
terraform {
  required_version = ">= 0.12"
}

provider "esxi" {
  esxi_hostname = "10.4.100.200"
  esxi_hostport = "22"
  esxi_hostssl  = "443"
  esxi_username = "root"
  esxi_password = "< password >"
}

# Input variable for the guest name
#variable "guest_name" {
#  description = "Name for the guest VM"
#}

# Create the VM
resource "esxi_guest" "vmtest" {
  guest_name      = "check-win10"
  disk_store      = "datastore1"
  ovf_source      = "/home/deployer/Backup/check-win10/check-
win10.ovf"

  # Setting memory size and number of CPUs to ensure VM has
enough resources
  memsize         = 2048    # 2 GB RAM
  numvcpus        = 2       # 2 CPUs

  # Specify EFI as the boot firmware type
  boot_firmware   = "efi"

  # Ensure the VM is powered on after creation
  power           = "on"

  network_interfaces {
    virtual_network = "VM Network"
  }

  # DHCP settings - guest_startup_timeout ensures enough time for
DHCP to assign an IP
  guest_startup_timeout = 60
}
```

Figure 34: *Windows Terraform Configuration*



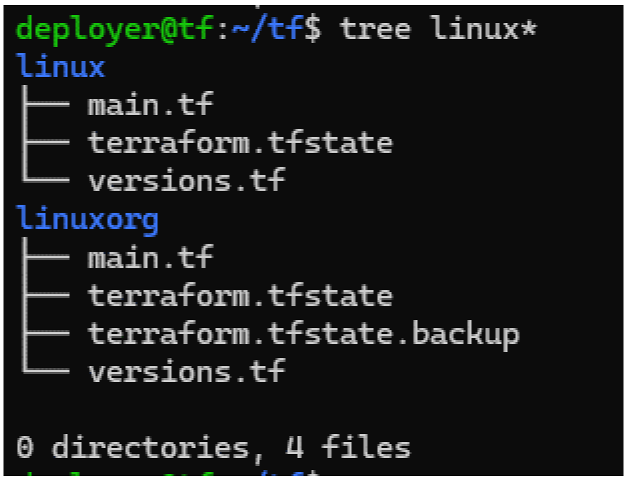
```
deployer@tf:~/tf$ tree windows*
windows
├── main.tf
├── terraform.tfstate
└── versions.tf
windowsorg
├── main.tf
├── terraform.tfstate
├── terraform.tfstate.backup
└── versions.tf

0 directories, 4 files
```

Figure 35: *Wnidows Terraform Configuration Folder*

5.2.2.2 Linux Machine Terraform Configuration:

```
terraform {
  required_version = ">= 0.12"
}

provider "esxi" {
  esxi_hostname = "10.4.100.200"
  esxi_hostport = "22"
  esxi_hostssl  = "443"
  esxi_username = "root"
  esxi_password = " <password> "
}

# Input variable for the guest name
#variable "guest_name" {
 # description = "Name for the guest VM"
#}

# Create the VM
resource "esxi_guest" "vmtest" {
  guest_name         = "check-linux"
  disk_store      = "datastore1"
  ovf_source      = "/home/deployer/Backup/check-linux/check-
linux.ovf"

  # Setting memory size and number of CPUs to ensure VM has
enough resources
  memsize         = 2048    # 2 GB RAM
  numvcpus        = 2       # 2 CPUs

  # Specify EFI as the boot firmware type
  boot_firmware   = "efi"
  # Ensure the VM is powered on after creation
  power            = "on"

  network_interfaces {
    virtual_network = "VM Network"
  }

  # DHCP settings - guest_startup_timeout ensures enough time for
DHCP to assign an IP
  guest_startup_timeout = 60
}
```

Figure 36: Ubuntu Terraform Configuration



```
deployer@tf:~/tf$ tree linux*
linux
├── main.tf
├── terraform.tfstate
└── versions.tf
linuxorg
├── main.tf
├── terraform.tfstate
├── terraform.tfstate.backup
└── versions.tf

0 directories, 4 files
```

Figure 37: *Windows Terraform Configuration Folder*

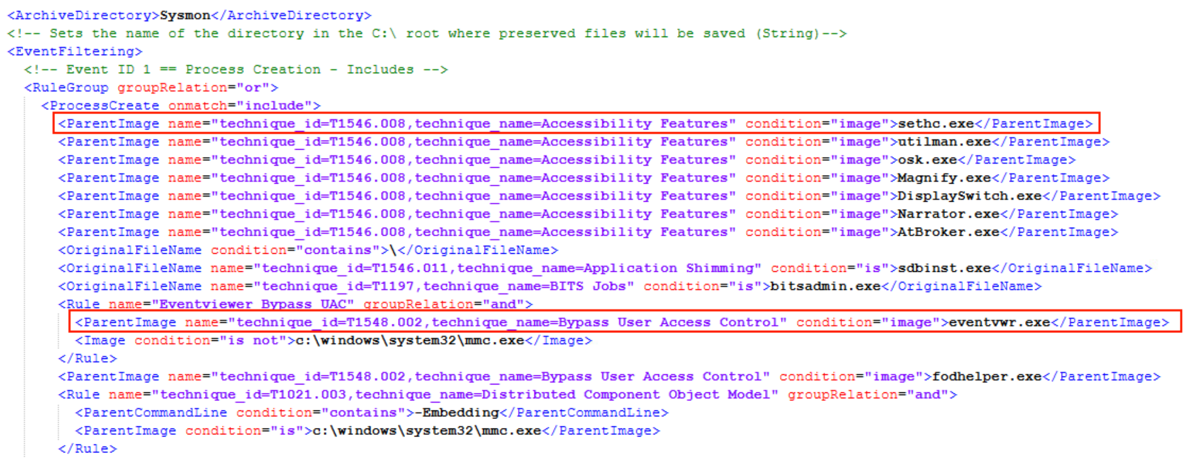
5.2.3 Event Management Tool Configuration with MITRE ATT&CK Framework

The Windows Event Management Tool Configuration is a critical aspect of Hive Sentinel, ensuring comprehensive logging and monitoring of system events across both Windows and Linux environments. Sysmon is employed for Windows systems, providing detailed logs of system activity to detect and analyze malicious actions. For Linux systems, auditd is used to track security-relevant events, ensuring thorough monitoring and analysis of potential threats. MITRE ATT&CK mapping is integrated into both Sysmon and auditd configurations by defining specific MITRE ATT&CK rules in their respective configuration files. This enables precise detection and categorization of tactics, techniques, and procedures (TTPs) used by adversaries.

5.2.3.1 Sysmon Configuration

Sysmon, a system monitoring tool from Microsoft, is configured to capture detailed event logs. This involves installing Sysmon on Windows machines and defining a configuration file that specifies the types of events to monitor, such as process creations, network connections, and file modifications. The configuration file also includes MITRE ATT&CK rules, allowing the detection and mapping of malicious activities to known ATT&CK techniques. The logs generated by Sysmon are then forwarded to a central logging database Elastic search for analysis.

Figure 38 illustrates the Sysmon Rule with MITRE ATT&CK mapping for technique name Accessibility Features and Bypass User Access Control



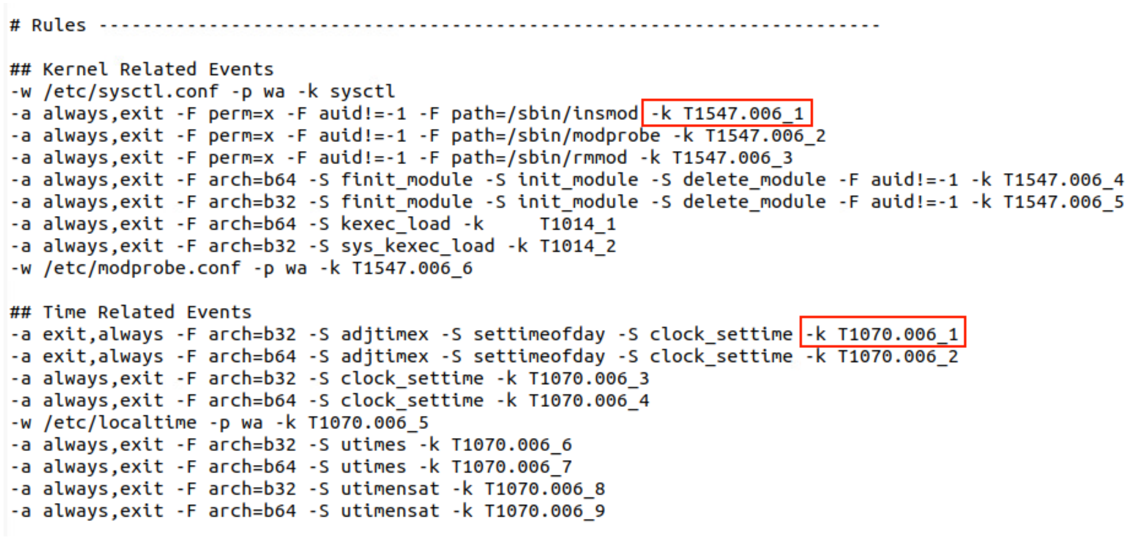
```
<ArchiveDirectory>Sysmon</ArchiveDirectory>
<!-- Sets the name of the directory in the C:\ root where preserved files will be saved (String)-->
<EventFiltering>
  <!-- Event ID 1 == Process Creation - Includes -->
  <RuleGroup groupRelation="or">
    <ProcessCreate onmatch="include">
      <ParentImage name="technique_id=T1546.008,technique_name=Accessibility Features" condition="image">sethc.exe</ParentImage>
      <ParentImage name="technique_id=T1546.008,technique_name=Accessibility Features" condition="image">utilman.exe</ParentImage>
      <ParentImage name="technique_id=T1546.008,technique_name=Accessibility Features" condition="image">osk.exe</ParentImage>
      <ParentImage name="technique_id=T1546.008,technique_name=Accessibility Features" condition="image">Magnify.exe</ParentImage>
      <ParentImage name="technique_id=T1546.008,technique_name=Accessibility Features" condition="image">DisplaySwitch.exe</ParentImage>
      <ParentImage name="technique_id=T1546.008,technique_name=Accessibility Features" condition="image">Narrator.exe</ParentImage>
      <ParentImage name="technique_id=T1546.008,technique_name=Accessibility Features" condition="image">AtBroker.exe</ParentImage>
      <OriginalFileName condition="contains">\</OriginalFileName>
      <OriginalFileName name="technique_id=T1546.011,technique_name=Application Shimming" condition="is">sdbinst.exe</OriginalFileName>
      <OriginalFileName name="technique_id=T1197,technique_name=BITS Jobs" condition="is">bitsadmin.exe</OriginalFileName>
      <Rule name="Eventviewer Bypass UAC" groupRelation="and">
        <ParentImage name="technique_id=T1548.002,technique_name=Bypass User Access Control" condition="image">eventvwr.exe</ParentImage>
        <Image condition="is not">c:\windows\system32\mmc.exe</Image>
      </Rule>
      <ParentImage name="technique_id=T1548.002,technique_name=Bypass User Access Control" condition="image">fodhelper.exe</ParentImage>
      <Rule name="technique_id=T1021.003,technique_name=Distributed Component Object Model" groupRelation="and">
        <ParentCommandLine condition="contains">-Embedding</ParentCommandLine>
        <ParentImage condition="is">c:\windows\system32\mmc.exe</ParentImage>
      </Rule>
```

Figure 38: *Sysmon configuration rules with MITRE ATT&CK mapping*

5.2.3.2 Auditd Configuration

Auditd, the Linux Audit Daemon, is configured to monitor and record security-relevant events on Linux systems. Configuration includes setting up audit rules that dictate which events to capture, such as file accesses, user logins, and system calls. The audit rules also incorporate MITRE ATT&CK techniques, enabling the precise mapping of observed activities to know adversarial behaviors. The audit logs are collected and sent to a central repository for analysis and correlation with other security data.

Figure 39 illustrates the Auditd Rule with MITRE ATT&CK mapping for technique id T1547.006 and T1070.006



```
# Rules ----------------------------------------------------------------------

## Kernel Related Events
-w /etc/sysctl.conf -p wa -k sysctl
-a always,exit -F perm=x -F auid!=-1 -F path=/sbin/insmod -k T1547.006_1
-a always,exit -F perm=x -F auid!=-1 -F path=/sbin/modprobe -k T1547.006_2
-a always,exit -F perm=x -F auid!=-1 -F path=/sbin/rmmod -k T1547.006_3
-a always,exit -F arch=b64 -S finit_module -S init_module -S delete_module -F auid!=-1 -k T1547.006_4
-a always,exit -F arch=b32 -S finit_module -S init_module -S delete_module -F auid!=-1 -k T1547.006_5
-a always,exit -F arch=b64 -S kexec_load -k     T1014_1
-a always,exit -F arch=b32 -S sys_kexec_load -k T1014_2
-w /etc/modprobe.conf -p wa -k T1547.006_6

## Time Related Events
-a exit,always -F arch=b32 -S adjtimex -S settimeofday -S clock_settime -k T1070.006_1
-a exit,always -F arch=b64 -S adjtimex -S settimeofday -S clock_settime -k T1070.006_2
-a always,exit -F arch=b32 -S clock_settime -k T1070.006_3
-a always,exit -F arch=b64 -S clock_settime -k T1070.006_4
-w /etc/localtime -p wa -k T1070.006_5
-a always,exit -F arch=b32 -S utimes -k T1070.006_6
-a always,exit -F arch=b64 -S utimes -k T1070.006_7
-a always,exit -F arch=b32 -S utimensat -k T1070.006_8
-a always,exit -F arch=b64 -S utimensat -k T1070.006_9
```

Figure 39: *Auditd configuration rules with MITRE ATT&CK mapping*

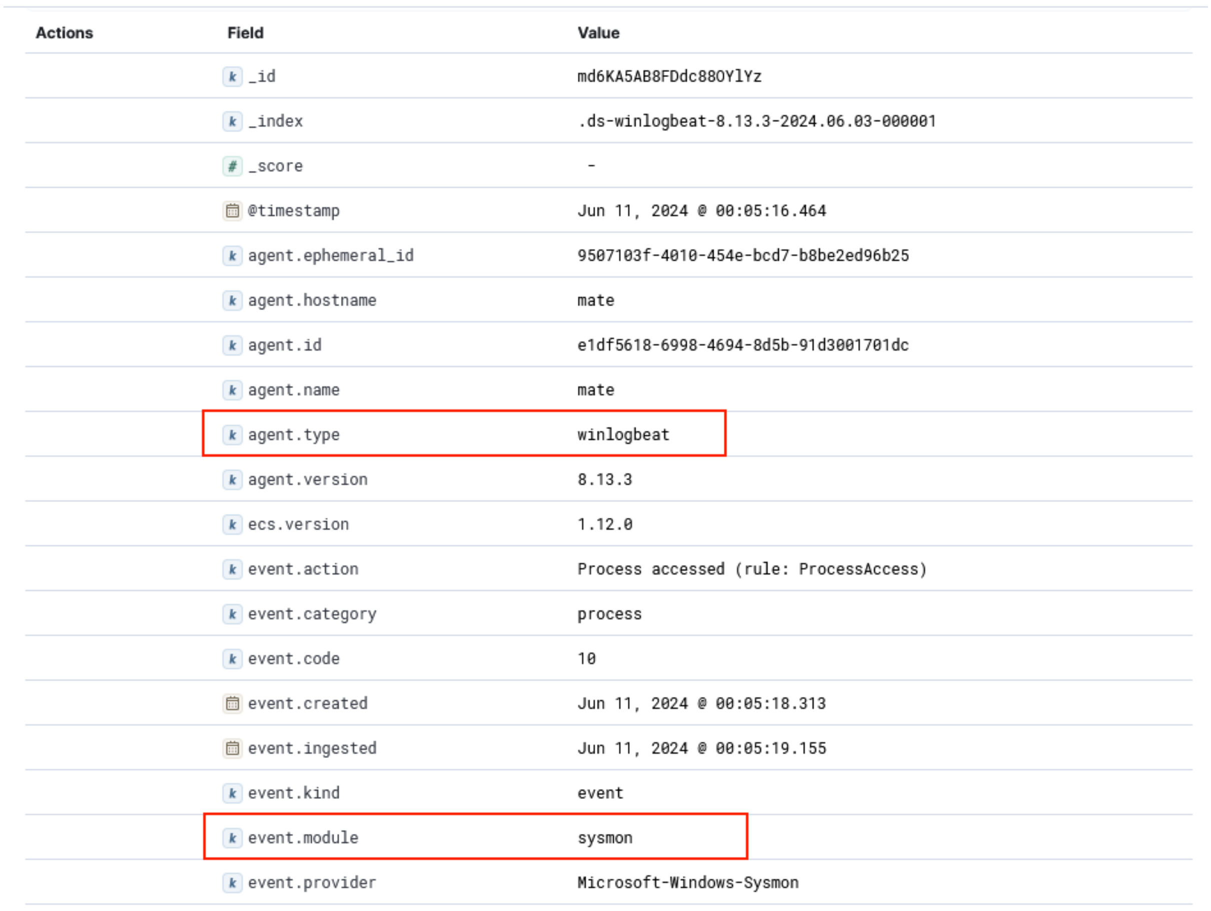
5.2.4 Elastic search Integration

Database integration is a crucial component of Hive Sentinel, facilitating the centralized collection, storage, and analysis of log data from various honeypots. Elasticsearch is employed for this purpose, offering a robust and scalable solution for handling large volumes of machine data. By aggregating logs from diverse sources into a single repository, Elasticsearch enables comprehensive monitoring and swift querying of security events across the entire infrastructure.

Elasticsearch collects log data from different honeypots in the form of index pattern files, which organize and structure the incoming data for efficient storage and retrieval. This centralized approach not only simplifies the management of logs but also enhances the ability to perform complex queries and analytics. With Elasticsearch, security analysts can quickly identify patterns, investigate incidents, and correlate data from multiple honeypots to gain deeper insights into potential threats and vulnerabilities within the network.

5.2.4.1 Winlogbeat:

Winlogbeat is a lightweight data shipper for Windows event logs, part of the Elastic Stack. It is designed to capture and forward Windows event logs to Elasticsearch or Logstash for further analysis and visualization. This makes it especially useful for monitoring and troubleshooting Windows environments, allowing for detailed insights into system and application events on Windows servers and workstations.

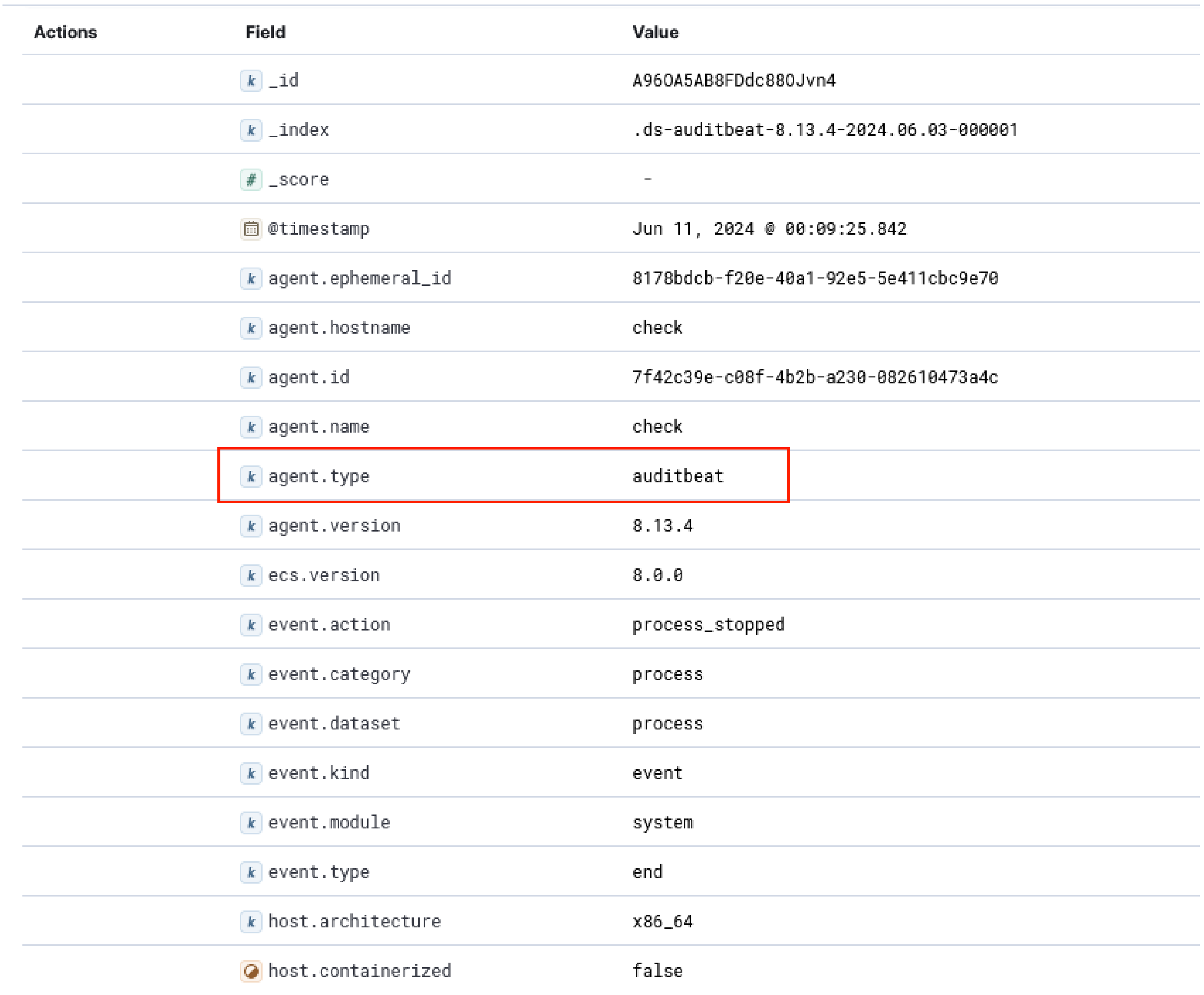


Actions	Field	Value
	_id	md6KA5AB8FDdc88OYlYz
	_index	.ds-winlogbeat-8.13.3-2024.06.03-000001
	_score	-
	@timestamp	Jun 11, 2024 @ 00:05:16.464
	agent.ephemeral_id	9507103f-4010-454e-bcd7-b8be2ed96b25
	agent.hostname	mate
	agent.id	e1df5618-6998-4694-8d5b-91d3001701dc
	agent.name	mate
	agent.type	winlogbeat
	agent.version	8.13.3
	ecs.version	1.12.0
	event.action	Process accessed (rule: ProcessAccess)
	event.category	process
	event.code	10
	event.created	Jun 11, 2024 @ 00:05:18.313
	event.ingested	Jun 11, 2024 @ 00:05:19.155
	event.kind	event
	event.module	sysmon
	event.provider	Microsoft-Windows-Sysmon

Figure 40: *Winlogbeat index pattern*

5.2.4.2 Auditbeat

Auditbeat is a data shipper focused on auditing activities within a system. It collects data on user logins, file integrity, process activities, and other security-related events. By forwarding this information to Elasticsearch or Logstash, Auditbeat helps organizations maintain security and compliance, providing detailed logs that are essential for security monitoring and forensic analysis.

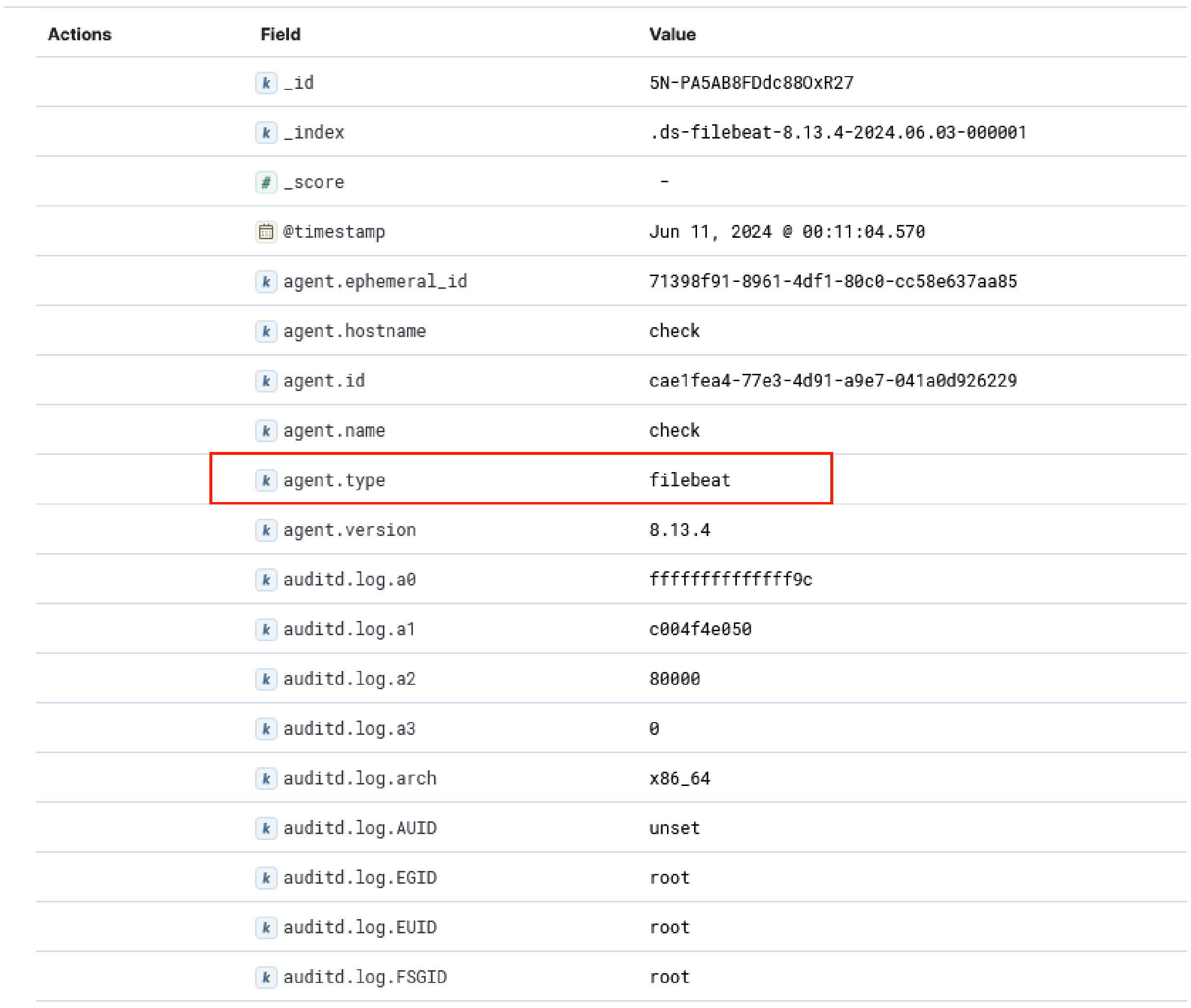


Actions	Field	Value
	_id	A96OA5AB8FDdc88OJvn4
	_index	.ds-auditbeat-8.13.4-2024.06.03-000001
	_score	-
	@timestamp	Jun 11, 2024 @ 00:09:25.842
	agent.ephemeral_id	8178bdcb-f20e-40a1-92e5-5e411cbc9e70
	agent.hostname	check
	agent.id	7f42c39e-c08f-4b2b-a230-082610473a4c
	agent.name	check
	agent.type	auditbeat
	agent.version	8.13.4
	ecs.version	8.0.0
	event.action	process_stopped
	event.category	process
	event.dataset	process
	event.kind	event
	event.module	system
	event.type	end
	host.architecture	x86_64
	host.containerized	false

Figure 41: *Auditbeat index pattern*

5.2.4.3 Filebeat

Filebeat is another member of the Elastic Stack, designed to collect and ship log data from various sources to Elasticsearch or Logstash. It is versatile and lightweight, capable of gathering logs from files, databases, and network services. Filebeat helps centralize log data, making it easier to manage, analyze, and visualize logs from diverse sources in a unified manner.

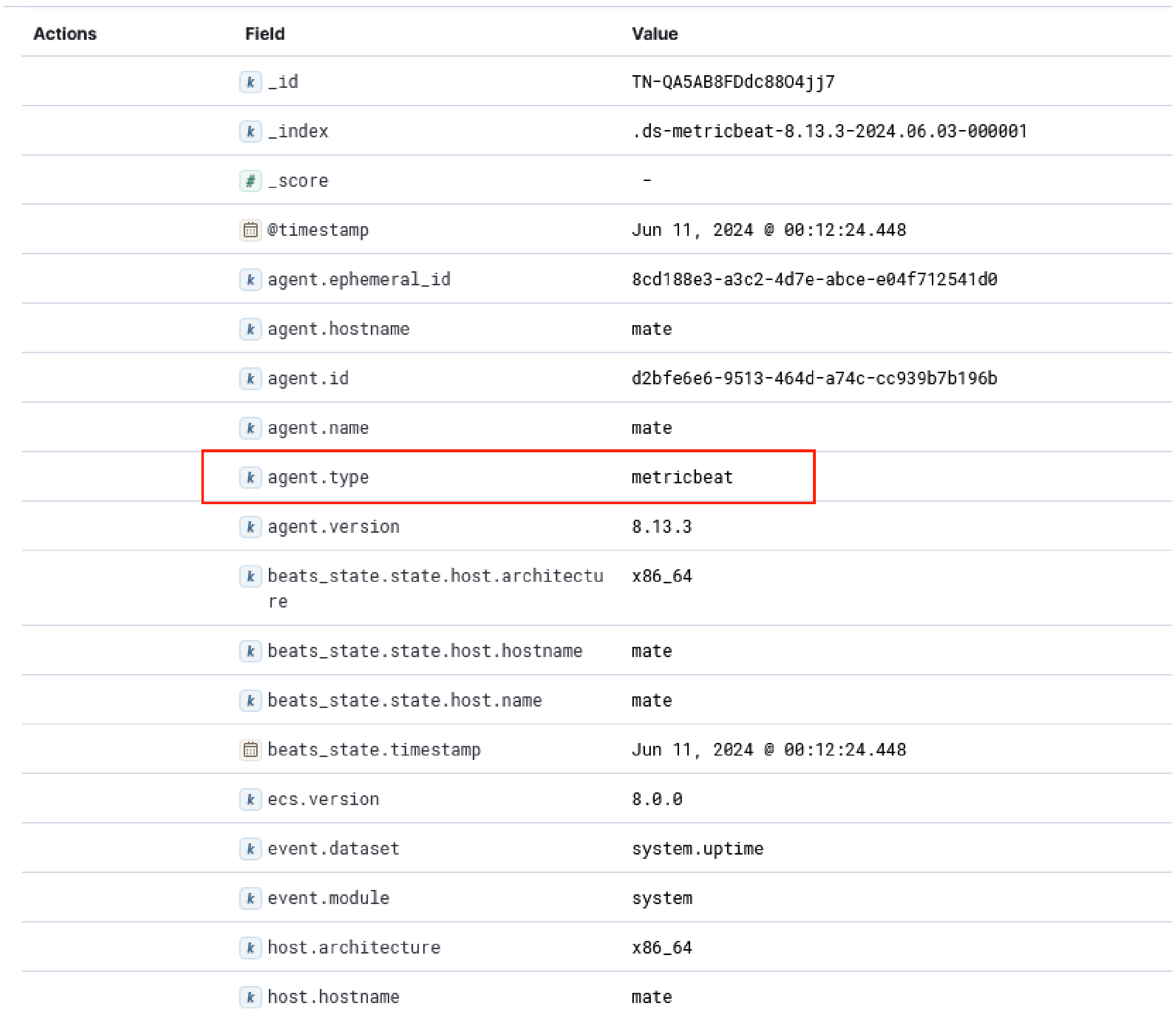


Actions	Field	Value
	_id	5N-PA5AB8FDdc88OxR27
	_index	.ds-filebeat-8.13.4-2024.06.03-000001
	_score	-
	@timestamp	Jun 11, 2024 @ 00:11:04.570
	agent.ephemeral_id	71398f91-8961-4df1-80c0-cc58e637aa85
	agent.hostname	check
	agent.id	cae1fea4-77e3-4d91-a9e7-041a0d926229
	agent.name	check
	agent.type	filebeat
	agent.version	8.13.4
	auditd.log.a0	ffffffffffffff9c
	auditd.log.a1	c004f4e050
	auditd.log.a2	80000
	auditd.log.a3	0
	auditd.log.arch	x86_64
	auditd.log.AUID	unset
	auditd.log.EGID	root
	auditd.log.EUID	root
	auditd.log.FSGID	root

Figure 42: *Filebeat index pattern*

5.2.4.4 Metricbeat

Metricbeat is a data shipper for collecting and shipping metrics from systems and services. It gathers vital data on resource usage, performance, and operational health from operating systems, databases, and applications. By sending this data to Elasticsearch or Logstash, Metricbeat enables detailed monitoring of infrastructure and application performance, helping to identify issues and optimize resource utilization.

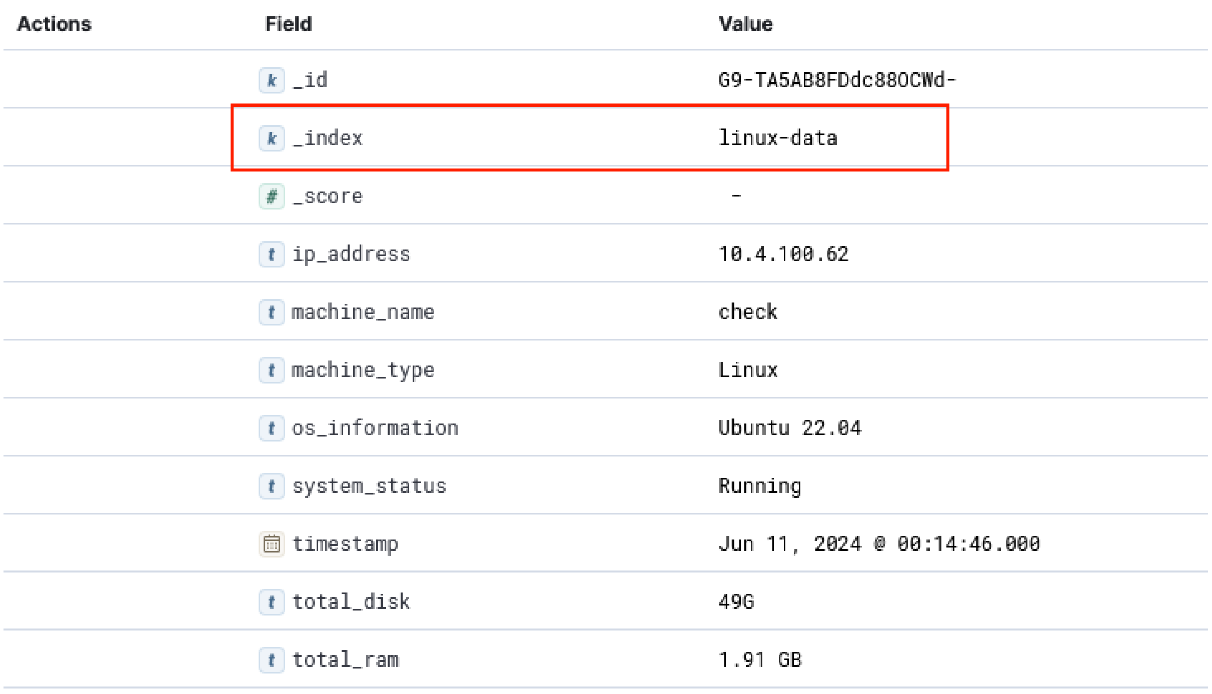


Actions	Field	Value
	_id	TN-QA5AB8FDdc8804jj7
	_index	.ds-metricbeat-8.13.3-2024.06.03-000001
	_score	-
	@timestamp	Jun 11, 2024 @ 00:12:24.448
	agent.ephemeral_id	8cd188e3-a3c2-4d7e-abce-e04f712541d0
	agent.hostname	mate
	agent.id	d2bfe6e6-9513-464d-a74c-cc939b7b196b
	agent.name	mate
	agent.type	metricbeat
	agent.version	8.13.3
	beats_state.state.host.architecture	x86_64
	beats_state.state.host.hostname	mate
	beats_state.state.host.name	mate
	beats_state.timestamp	Jun 11, 2024 @ 00:12:24.448
	ecs.version	8.0.0
	event.dataset	system.uptime
	event.module	system
	host.architecture	x86_64
	host.hostname	mate

Figure 43: *Metricbeat index pattern*

5.2.4.5 LinuxDataBeat

LinuxDataBeat is a custom Beat designed to collect and forward detailed information about Linux systems to Elasticsearch. This Beat gathers data such as the timestamp, machine name, system status, machine type, operating system information, IP address, total RAM, and total disk space. For example, it can capture information like a machine named "check" running Ubuntu 22.04 with 1.91 GB of RAM and a total disk space of 49 GB. The collected data helps in monitoring the status and resource usage of Linux machines within an IT infrastructure, providing valuable insights for system administrators and IT operations teams.

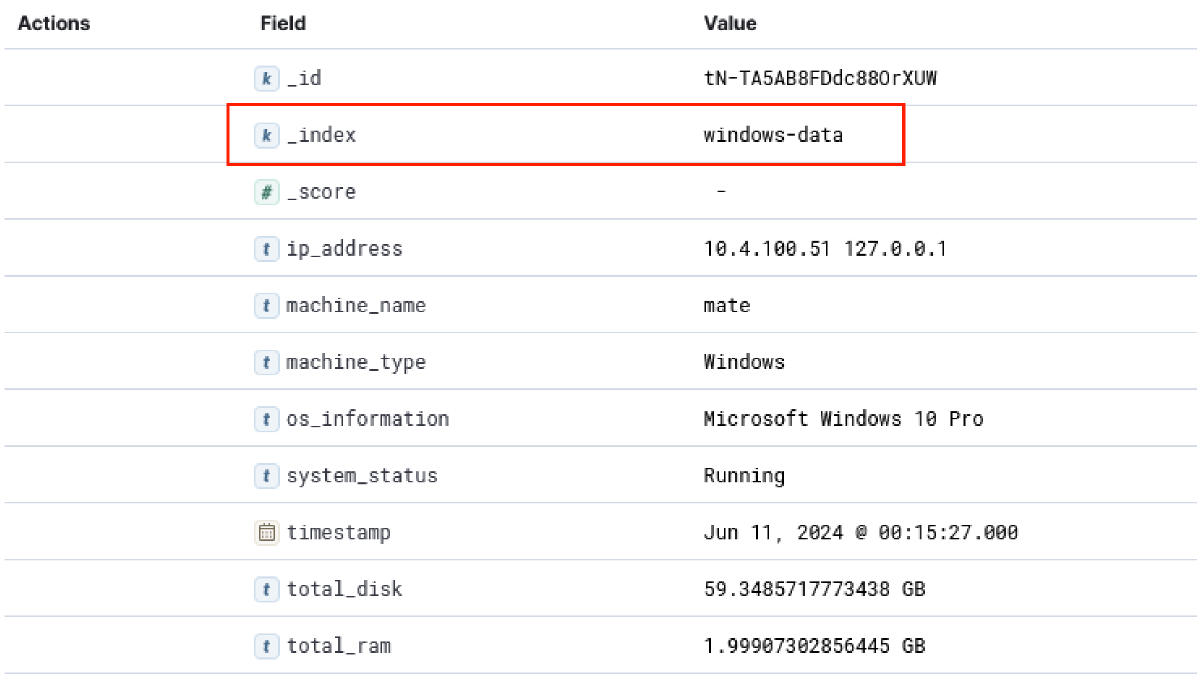


Actions	Field	Value
	_id	G9-TA5AB8FDdc88OCWd-
	_index	linux-data
	_score	-
	ip_address	10.4.100.62
	machine_name	check
	machine_type	Linux
	os_information	Ubuntu 22.04
	system_status	Running
	timestamp	Jun 11, 2024 @ 00:14:46.000
	total_disk	49G
	total_ram	1.91 GB

Figure 44: *linux-data index pattern*

5.2.4.6 WindowsDataBeat

WindowsDataBeat is another custom Beat tailored for Windows systems. It collects and sends data about Windows machines to Elasticsearch, including the timestamp, machine name, system status, machine type, operating system details, IP addresses, total RAM, and total disk space. An example of data captured by this Beat includes a machine named "mate" running Microsoft Windows 10 Pro with 1.999 GB of RAM and 59.35 GB of disk space. This Beat is instrumental in monitoring and managing Windows environments, ensuring that system health and resource utilization are continuously tracked and analyzed.



Actions	Field	Value
	k _id	tN-TA5AB8FDdc88OrXUW
	k _index	windows-data
	# _score	-
	t ip_address	10.4.100.51 127.0.0.1
	t machine_name	mate
	t machine_type	Windows
	t os_information	Microsoft Windows 10 Pro
	t system_status	Running
	timestamp	Jun 11, 2024 @ 00:15:27.000
	t total_disk	59.3485717773438 GB
	t total_ram	1.99907302856445 GB

Figure 45: *windows-datat index pattern*

Chapter 6

Results & Discussion

6. Introduction

In this chapter, the implementation of the Health Monitoring and Control Management (HMCM) methodology for honeypots, specifically the Hive Sentinel, is presented along with a proof of concept to demonstrate real-time detection and response to malicious activities. The proof of concept involves attack scenarios conducted in a controlled lab environment, focusing on the detection capabilities of the HMCM system. The analysis aims to answer the following questions:

1. How to detect an initial compromise by command injection?
2. How to detect different adversaries interacting with the system?
3. Automated deployment of honeypots with Terraform
4. Health monitoring snapshot restoration

6.1 How to Detect Initial Compromise

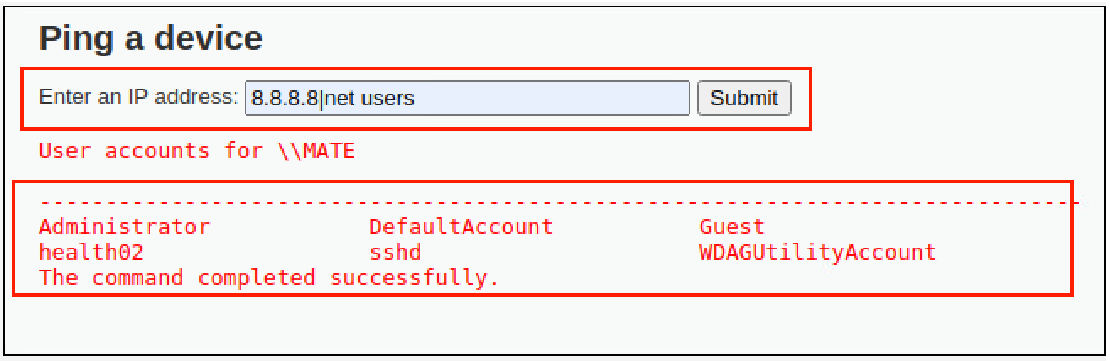
6.1.1 Command Injection Example: 8.8.8.8|net users

In this example, we consider a simple command injection attack using 8.8.8.8|net users. This type of attack exploits a vulnerable web application by injecting commands that are executed by the system shell.

Step-by-Step Explanation:

1. **Vulnerability Identification:** The attacker identifies a command injection vulnerability in a web form or URL parameter of the target application.
2. **Payload Injection:** The attacker crafts a payload, 8.8.8.8|net users, and submits it through the vulnerable input field. Here, 8.8.8.8 is a benign input, and |net users is the malicious part that gets executed.
3. **Execution:** The server processes the input, and due to improper input sanitization, the command net users get executed in the system shell.
4. **Response:** The server responds with the result of the net users command, typically revealing the user under which the server process is running.

Test Scenario 1: Command injection on DVWA



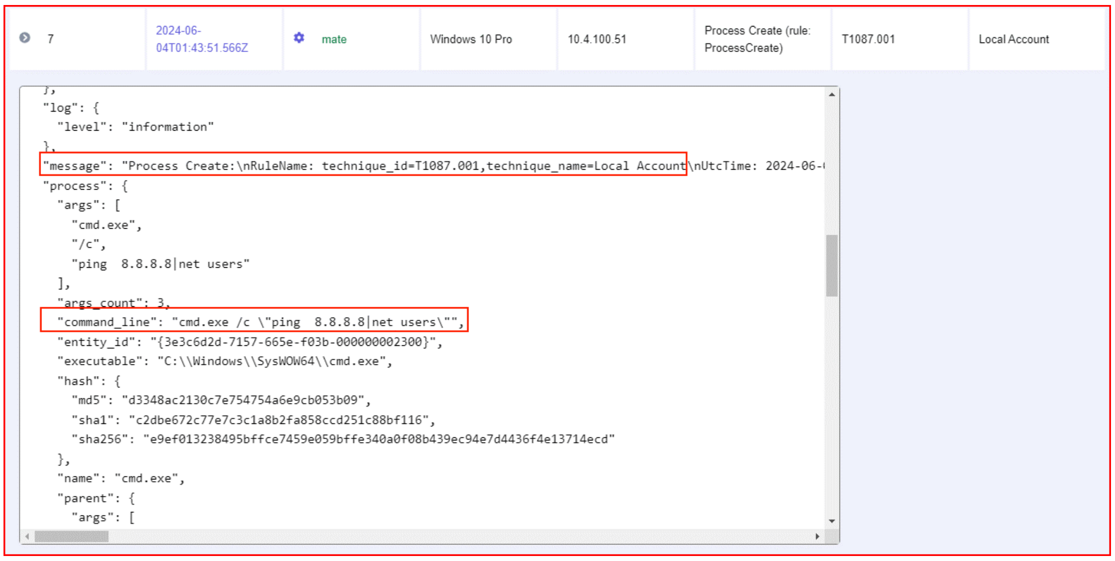
Ping a device

Enter an IP address: 8.8.8.8|net users [Submit]

```
User accounts for \\MATE

-------------------------------------------------------------------------------
Administrator            DefaultAccount           Guest
health02                 sshd                     WDAGUtilityAccount
The command completed successfully.
```

Figure 46: *Command injection attack on DVWA*

Detection Using Dashboard: To detect such initial compromises, we can set up Elasticsearch to monitor and log command execution events. A query to detect this specific attack vector might look like this on our own dashboard:



7	2024-06-04T01:43:51.566Z	mate	Windows 10 Pro	10.4.100.51	Process Create (rule: ProcessCreate)	T1087.001	Local Account

```
  },
  "log": {
    "level": "information"
  },
  "message": "Process Create:\nRuleName: technique_id=T1087.001,technique_name=Local Account\nUtcTime: 2024-06-
  "process": {
    "args": [
      "cmd.exe",
      "/c",
      "ping  8.8.8.8|net users"
    ],
    "args_count": 3,
    "command_line": "cmd.exe /c \"ping  8.8.8.8|net users\"",
    "entity_id": "{3e3c6d2d-7157-665e-f03b-000000002300}",
    "executable": "C:\\Windows\\SysWOW64\\cmd.exe",
    "hash": {
      "md5": "d3348ac2130c7e754754a6e9cb053b09",
      "sha1": "c2dbe672c77e7c3c1a8b2fa858ccd251c88bf116",
      "sha256": "e9ef013238495bffce7459e059bffe340a0f08b439ec94e7d4436f4e13714ecd"
    },
    "name": "cmd.exe",
    "parent": {
      "args": [
```

Figure 47: *Log creation of Command injection attack on DVWA*

This query checks for:

- Parent process related to HTTP (indicating a web server process).
- Execution of cmd.exe (command prompt).
- Specific command line containing net users.

6.2 How to Detect Different Adversaries Interacting with the System

To identify different adversaries interacting with the system, the HMCM (Honeypot Monitoring and Control Mechanism) can enumerate all IP addresses that access the honeypot, treating each unique IP as a potential adversary.

Atomic Red Team Example:

Using Atomic Red Team, we can simulate adversary behavior for detection testing. Running atomic tests that simulate common adversary techniques helps validate our detection mechanisms.

PowerShell Script for Automated Testing and Cleanup

PowerShell script

```
function Run-AtomicRedTeamTests {
    Write-Host "Running Atomic Red Team tests..."

    $Tests = @(
        "T1082",     # System Information Discovery
        "T1012",     # Query Registry
        "T1124",     # System Time Discovery
        "T1047",     # Windows Management Instrumentation
        "T1033",     # System Owner/User Discovery
        "T1083"      # File and Directory Discovery
    )

    foreach ($Test in $Tests) {
        try {
            Invoke-AtomicTest $Test -TestNumbers 1
        } catch {
            Write-Error "Failed to run test $($Test): $_"
        }
    }

    Write-Host "Atomic Red Team tests completed."
}

function Cleanup-AtomicRedTeamTests {
    Write-Host "Cleaning up after Atomic Red Team tests..."

    $Tests = @(
        "T1082",
        "T1012",
        "T1124",
        "T1047",
        "T1033",
        "T1083"
    )

    foreach ($Test in $Tests) {
        try {
            Write-Host "Cleaning up after test $Test..."
            Invoke-AtomicTest $Test -Cleanup
        } catch {
            Write-Error "Failed to clean up test $($Test): $_"
        }
    }

    Write-Host "Cleanup of Atomic Red Team tests completed."
}

function Setup-AtomicRedTeamEnvironment {
    Write-Host "Setting up Atomic Red Team environment..."

    try {
        IEX (IWR
'https://raw.githubusercontent.com/redcanaryco/invoke-
atomicredteam/master/install-atomicredteam.ps1' -UseBasicParsing);
        Install-AtomicRedTeam -getAtomics -Force

        Run-AtomicRedTeamTests

        Cleanup-AtomicRedTeamTests
    } catch {
        Write-Error "Failed to setup Atomic Red Team environment:
$_"
        return
    }
}

Setup-AtomicRedTeamEnvironment
```

- **Run-AtomicRedTeamTests Function:**
    1. Initializes by displaying a message indicating the start of Atomic Red Team tests.
    2. Defines an array **$Tests** containing the names of specific Atomic Red Team tests to be executed.
    3. Executes each test in the **$Tests** array using the **Invoke-AtomicTest** cmdlet with the **-TestNumbers 1** parameter to run only one instance of each test.

4. Catches any errors that occur during the execution of individual tests and displays a corresponding error message.
5. Upon completion of all tests, displays a message confirming the completion of Atomic Red Team tests.

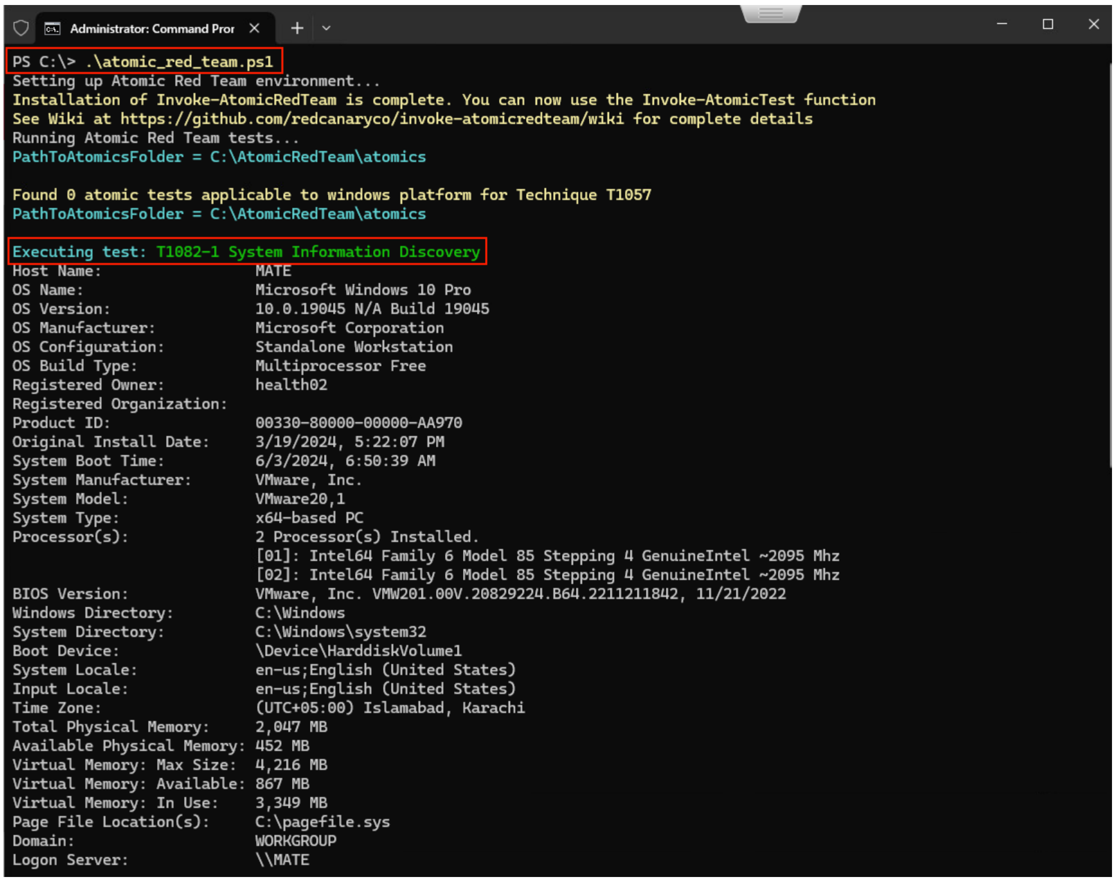
- **Cleanup-AtomicRedTeamTests Function:**
    1. Initializes by displaying a message indicating the start of cleanup after Atomic Red Team tests.
    2. Defines an array **$Tests** containing the names of Atomic Red Team tests for which cleanup actions need to be performed.
    3. Executes cleanup actions for each test in the **$Tests** array using the **Invoke-AtomicTest** cmdlet with the **-Cleanup** parameter.
    4. Catches any errors that occur during the cleanup process and displays a corresponding error message.
    5. Upon completion of all cleanup actions, displays a message confirming the completion of cleanup after Atomic Red Team tests.
- **Setup-AtomicRedTeamEnvironment Function:**
    1. Initializes by displaying a message indicating the start of setting up the Atomic Red Team environment.
    2. Attempts to install Atomic Red Team by executing a series of commands within a **try** block:
        - Invokes the **IEX** (Invoke-Expression) cmdlet to download and execute the script **install-atomicredteam.ps1** from the specified URL using **Invoke-WebRequest**.
        - Installs Atomic Red Team using the **Install-AtomicRedTeam** function with the **-getAtomics -Force** parameters, ensuring that the latest version is installed and any existing installation is forcefully replaced.
        - Calls the **Run-AtomicRedTeamTests** function to execute the predefined Atomic Red Team tests.

        - Calls the **Cleanup-AtomicRedTeamTests** function to perform cleanup actions after executing the tests.

    3. Catches any errors that occur during the setup process and displays a corresponding error message.
    4. Returns from the function if an error occurs, preventing subsequent actions.
    5. Upon successful completion of the setup process, displays a message confirming the setup of the Atomic Red Team environment.

- **Script Execution:**
    1. Initiates the execution of the **Setup-AtomicRedTeamEnvironment** function to set up the Atomic Red Team environment, execute the tests, and perform cleanup actions.
    2. Any errors encountered during the execution are caught and corresponding error messages are displayed.
    3. The script terminates after completing the setup, test execution, and cleanup processes.

Execution of script:

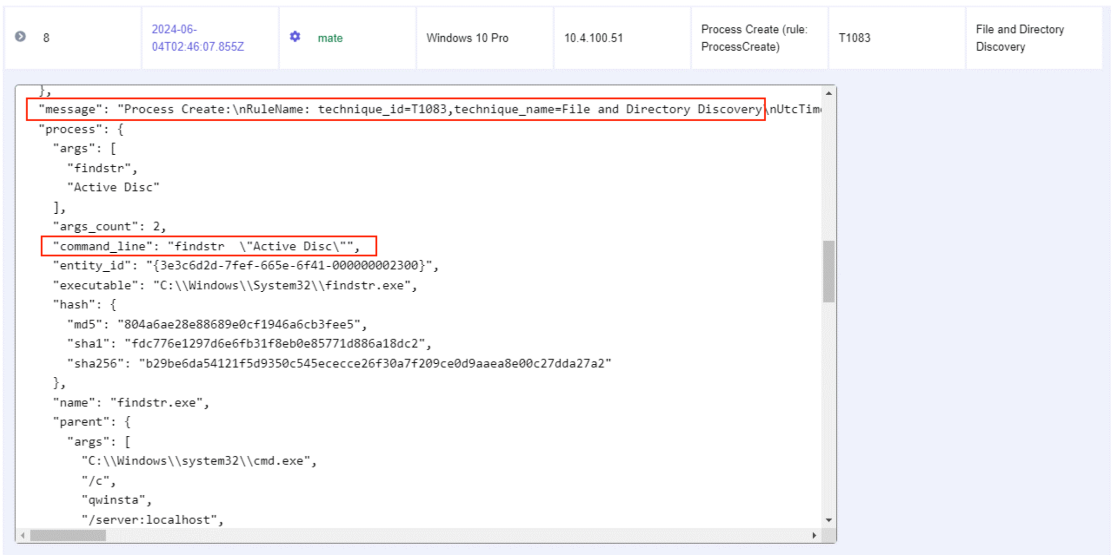


```
PS C:\> .\atomic_red_team.ps1
Setting up Atomic Red Team environment...
Installation of Invoke-AtomicRedTeam is complete. You can now use the Invoke-AtomicTest function
See Wiki at https://github.com/redcanaryco/invoke-atomicredteam/wiki for complete details
Running Atomic Red Team tests...
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Found 0 atomic tests applicable to windows platform for Technique T1057
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing test: T1082-1 System Information Discovery
Host Name:                 MATE
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.19045 N/A Build 19045
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          health02
Registered Organization:
Product ID:                00330-80000-00000-AA970
Original Install Date:     3/19/2024, 5:22:07 PM
System Boot Time:          6/3/2024, 6:50:39 AM
System Manufacturer:       VMware, Inc.
System Model:              VMware20,1
System Type:               x64-based PC
Processor(s):              2 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 85 Stepping 4 GenuineIntel ~2095 Mhz
                           [02]: Intel64 Family 6 Model 85 Stepping 4 GenuineIntel ~2095 Mhz
BIOS Version:              VMware, Inc. VMW201.00V.20829224.B64.2211211842, 11/21/2022
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+05:00) Islamabad, Karachi
Total Physical Memory:     2,047 MB
Available Physical Memory: 452 MB
Virtual Memory: Max Size:  4,216 MB
Virtual Memory: Available: 867 MB
Virtual Memory: In Use:    3,349 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\MATE
```

Figure 48: *Atomic red team cyber-attacks on windows 10 pro*

Detection Using Dashboard: To detect such initial compromises, we can set up Elasticsearch to monitor and log command execution events. A query to detect this specific attack vector might look like this on our own dashboard:

Test Scenario 2: Log creation of technique File and Directory Discovery after atomic red team attack on windows 10 pro

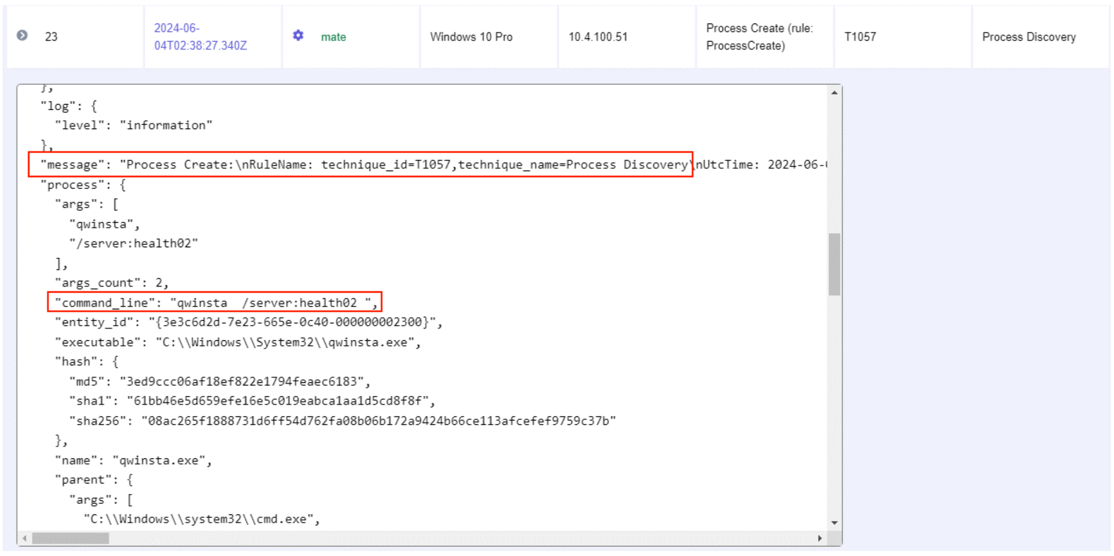


8	2024-06-04T02:46:07.855Z	mate	Windows 10 Pro	10.4.100.51	Process Create (rule: ProcessCreate)	T1083	File and Directory Discovery

```
  },
  "message": "Process Create:\nRuleName: technique_id=T1083,technique_name=File and Directory Discovery\nUtcTim
  "process": {
    "args": [
      "findstr",
      "Active Disc"
    ],
    "args_count": 2,
    "command_line": "findstr  \"Active Disc\"",
    "entity_id": "{3e3c6d2d-7fef-665e-6f41-000000002300}",
    "executable": "C:\\Windows\\System32\\findstr.exe",
    "hash": {
      "md5": "804a6ae28e88689e0cf1946a6cb3fee5",
      "sha1": "fdc776e1297d6e6fb31f8eb0e85771d886a18dc2",
      "sha256": "b29be6da54121f5d9350c545ececce26f30a7f209ce0d9aaea8e00c27dda27a2"
    },
    "name": "findstr.exe",
    "parent": {
      "args": [
        "C:\\Windows\\system32\\cmd.exe",
        "/c",
        "qwinsta",
        "/server:localhost",
```

Figure 49: *Log creation of File and Directory Discovery*

Test Scenario 3: Identifying technique Process Discovery after atomic red team attack on windows 10 Pro



23	2024-06-04T02:38:27.340Z	mate	Windows 10 Pro	10.4.100.51	Process Create (rule: ProcessCreate)	T1057	Process Discovery

```
  },
  "log": {
    "level": "information"
  },
  "message": "Process Create:\nRuleName: technique_id=T1057,technique_name=Process Discovery\nUtcTime: 2024-06-
  "process": {
    "args": [
      "qwinsta",
      "/server:health02"
    ],
    "args_count": 2,
    "command_line": "qwinsta  /server:health02 ",
    "entity_id": "{3e3c6d2d-7e23-665e-0c40-000000002300}",
    "executable": "C:\\Windows\\System32\\qwinsta.exe",
    "hash": {
      "md5": "3ed9ccc06af18ef822e1794feaec6183",
      "sha1": "61bb46e5d659efe16e5c019eabca1aa1d5cd8f8f",
      "sha256": "08ac265f1888731d6ff54d762fa08b06b172a9424b66ce113afcefef9759c37b"
    },
    "name": "qwinsta.exe",
    "parent": {
      "args": [
        "C:\\Windows\\system32\\cmd.exe",
```

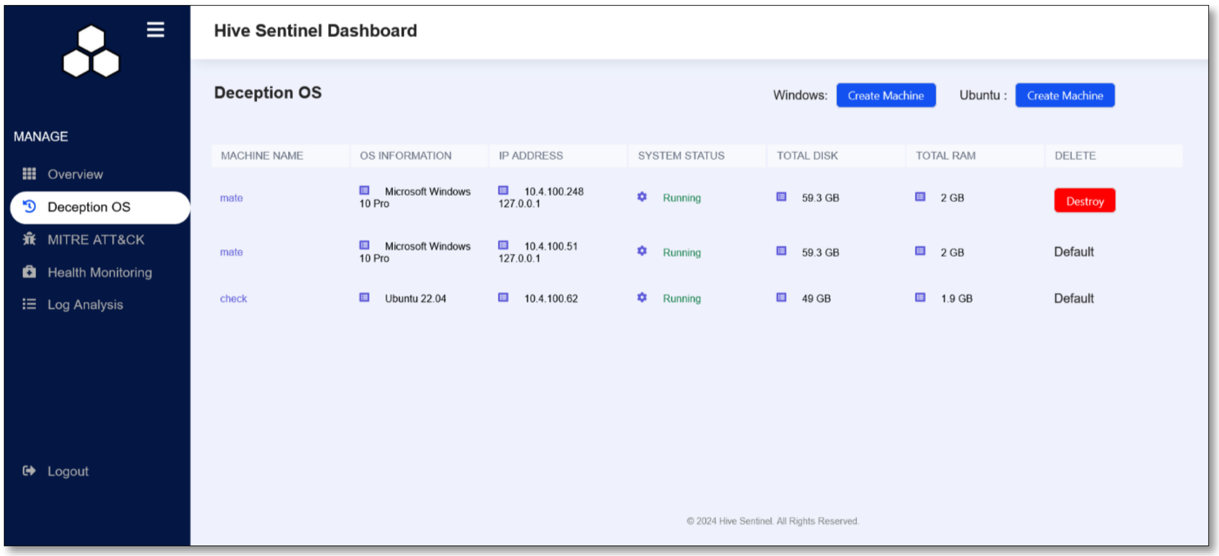
Figure 50: *Log creation of Process Discovery*

6.3 Automated Deployment of Honeypots with Terraform

Hive Sentinel can rapidly and consistently deploy high-interaction honeypots across various environments. Terraform scripts define the desired state of the infrastructure, enabling seamless provisioning and configuration of honeypots. This automation reduces manual intervention, minimizes deployment errors, and ensures that honeypots are consistently configured to the exact specifications required for effective deception. The automated deployment process not only accelerates the setup of these decoy systems but also ensures their continuous availability and optimal performance, thereby enhancing the overall efficacy of the deception platform.

6.3.1 Deception OS page (Dashboard)

The Hive Sentinel user interface simplifies honeypot deployment with a "Create Machines" button at the top right corner, offering "Windows" and "Ubuntu" options. Clicking either option triggers the execution of corresponding Terraform scripts, which automatically deploy and configure high-interaction honeypots on an ESXi hypervisor within six minutes. This rapid, automated process ensures quick and consistent deployment, enhancing the efficiency and responsiveness of the deception platform.



Hive Sentinel Dashboard

Deception OS

MACHINE NAME	OS INFORMATION	IP ADDRESS	SYSTEM STATUS	TOTAL DISK	TOTAL RAM	DELETE
mate	Microsoft Windows 10 Pro	10.4.100.248 127.0.0.1	Running	59.3 GB	2 GB	Destroy
mate	Microsoft Windows 10 Pro	10.4.100.51 127.0.0.1	Running	59.3 GB	2 GB	Default
check	Ubuntu 22.04	10.4.100.62	Running	49 GB	1.9 GB	Default

© 2024 Hive Sentinel. All Rights Reserved.

Figure 51: *Deception OS page*

Test Scenario 4: Creating Linux Honeypot named check-vmtest

Figure 52 illustrates the honeypot distribution before creating check-vmtest

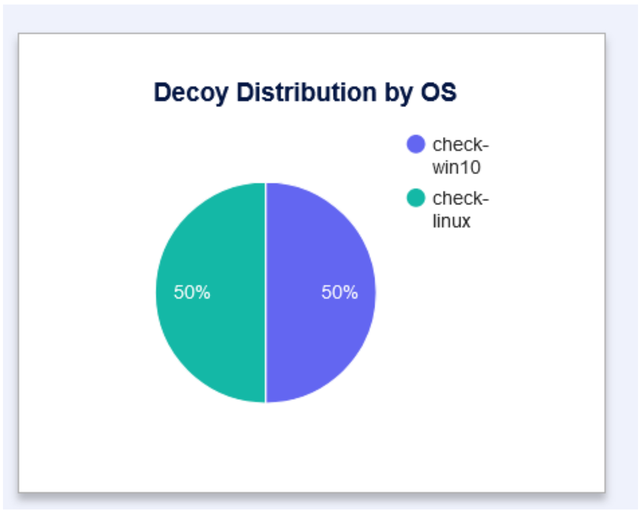
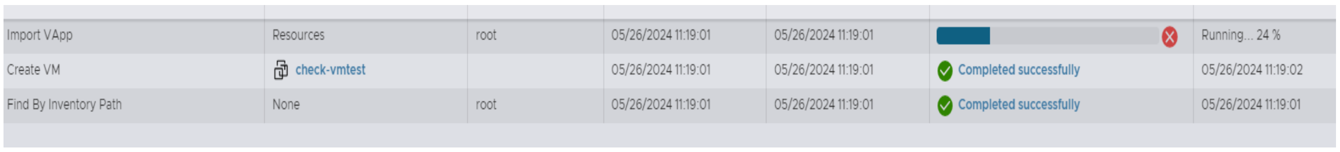


Figure 52: *Decoy Distribution by OS before creating machine*

Figure 53 illustrates the creation of check-vmtest on ESXI after clicking on Linux create machine button on deception OS page



Import VApp	Resources	root	05/26/2024 11:19:01	05/26/2024 11:19:01		Running... 24 %
Create VM	check-vmtest		05/26/2024 11:19:01	05/26/2024 11:19:01	✓ Completed successfully	05/26/2024 11:19:02
Find By Inventory Path	None	root	05/26/2024 11:19:01	05/26/2024 11:19:01	✓ Completed successfully	05/26/2024 11:19:01

Figure 53: *Creating check-vmtest linux honeypot*

Figure 54 illustrates the honeypot distribution after creating check-vmtest

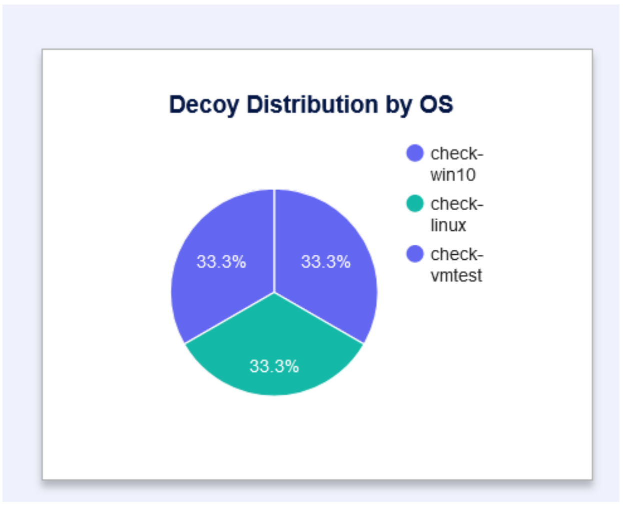
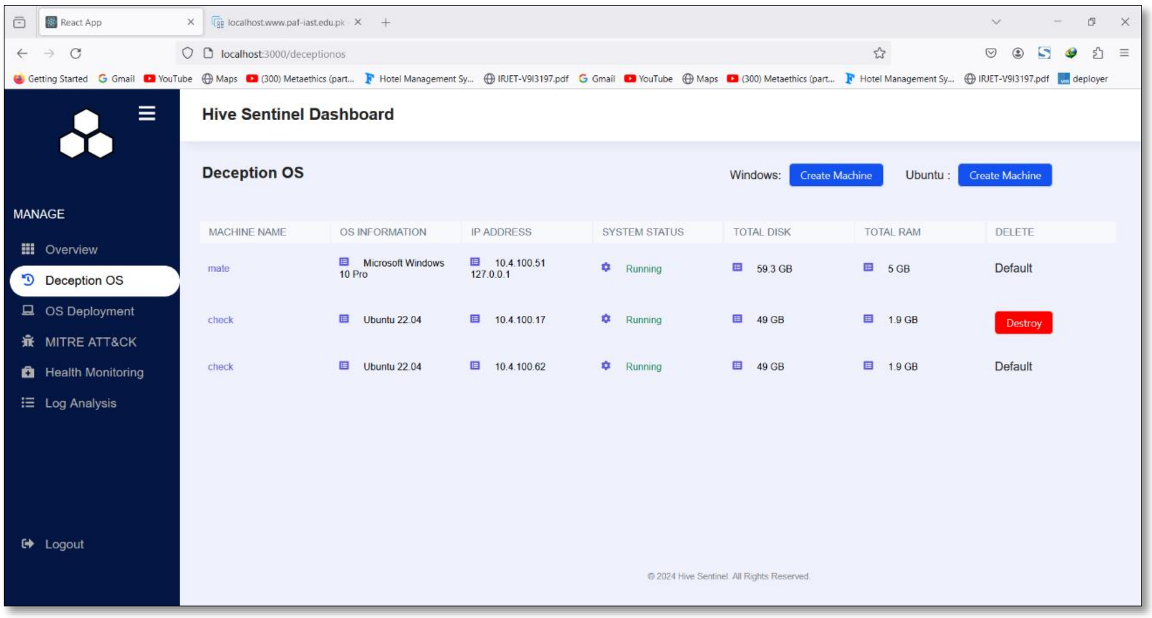


Figure 54: *check-vmtest linux honeypot status update on Dashboard*

Figure 55 illustrates the information of check-vmtest honeypot on Deception OS page



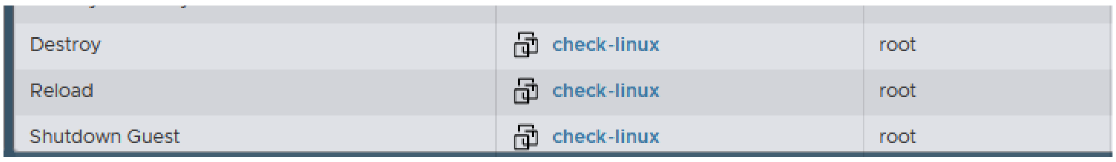
Deception OS

MACHINE NAME	OS INFORMATION	IP ADDRESS	SYSTEM STATUS	TOTAL DISK	TOTAL RAM	DELETE
mate	Microsoft Windows 10 Pro	10.4.100.51 127.0.0.1	Running	59.3 GB	5 GB	Default
check	Ubuntu 22.04	10.4.100.17	Running	49 GB	1.9 GB	Destroy
check	Ubuntu 22.04	10.4.100.62	Running	49 GB	1.9 GB	Default

Figure 55: *check-vmtest linux honeypot status update on Dashboard*

Test Scenario 5: Deleting honeypot check-linux

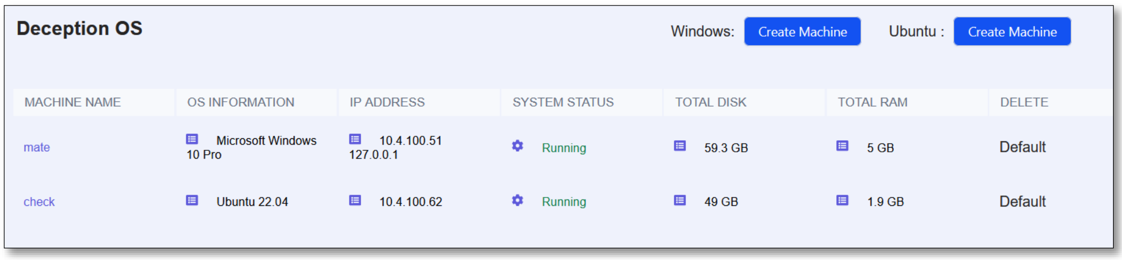
Figure 56 illustrates the deletion of check-linux machine after clicking destroy button on Desception OS page of respective honeypot



Destroy	check-linux	root
Reload	check-linux	root
Shutdown Guest	check-linux	root

Figure 56: *Deleting check-linux honeypot*

Figure 57 illustrates the information of default honeypots after deleting check-linux honeypot



Deception OS

Windows: Create Machine   Ubuntu : Create Machine

MACHINE NAME	OS INFORMATION	IP ADDRESS	SYSTEM STATUS	TOTAL DISK	TOTAL RAM	DELETE
mate	Microsoft Windows 10 Pro	10.4.100.51 127.0.0.1	Running	59.3 GB	5 GB	Default
check	Ubuntu 22.04	10.4.100.62	Running	49 GB	1.9 GB	Default

Figure 57: *Deception OS page information after deleting check-linux honeypot*

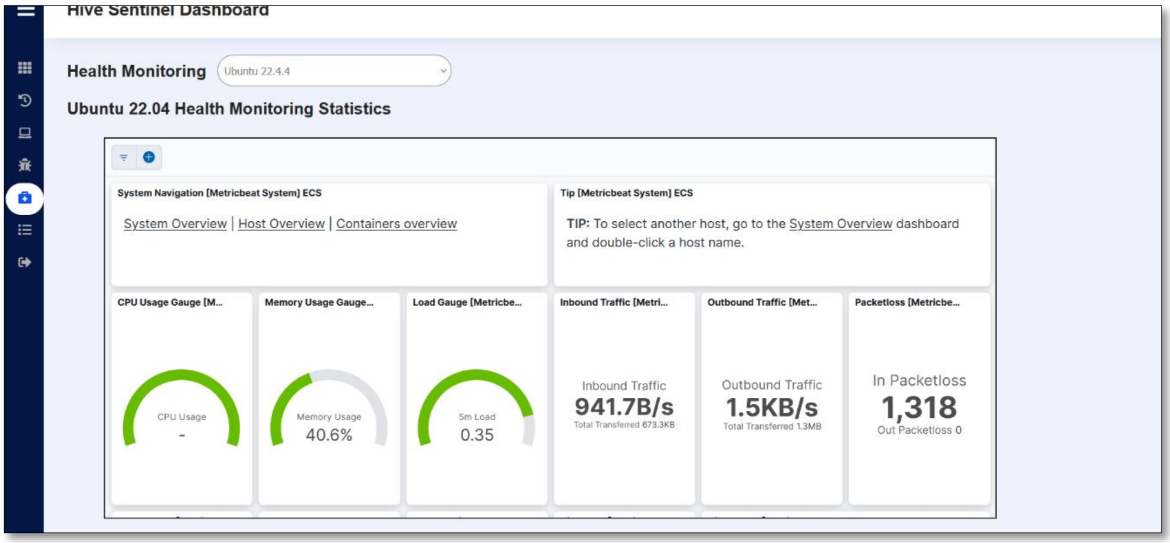
6.4 Health Monitoring Snapshot Restoration

In the Hive Sentinel project, health monitoring and snapshot restoration are critical components that ensure the system's effectiveness and resilience. Health monitoring involves continuously collecting and analyzing metrics such as CPU time and memory usage to detect anomalies or potential issues. Snapshot restoration complements this by enabling the system to quickly revert to a previously healthy state if any problems are detected.

Hive Sentinel continuously collects health monitoring parameters every minute and automatically compares them with predefined health thresholds. If any values exceed these thresholds, snapshots are restored using Terraform configuration scripts, ensuring the honeypots remain operational and effective.

Test Scenario 6: Snapshot restoration of compromised honeypot ubuntu 22.04

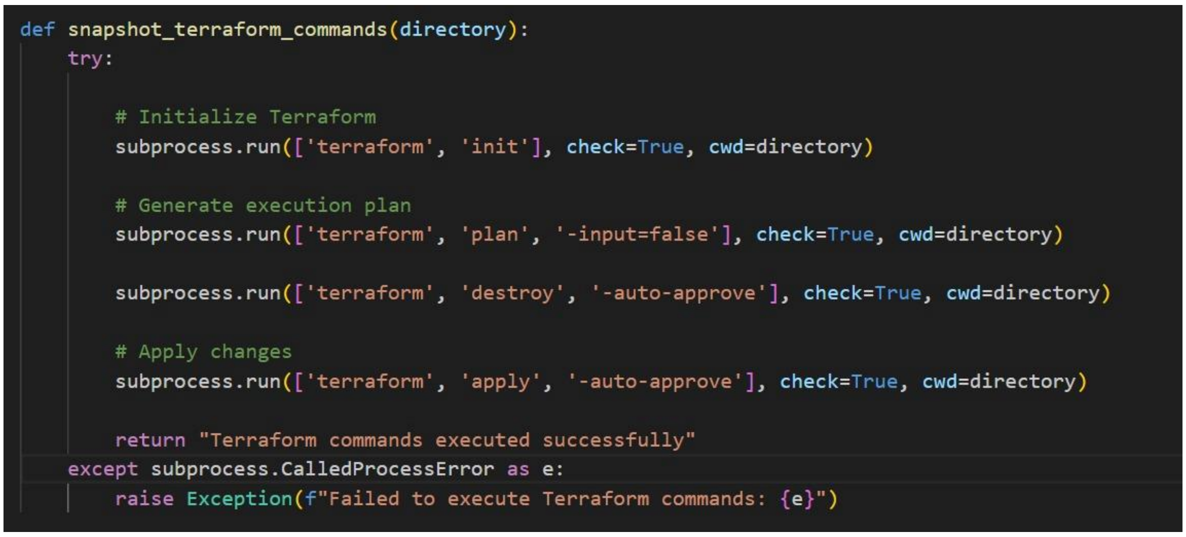
Figure 58 illustrates the health monitoring stats of compromised honeypot ubuntu 22.04



TIP: To select another host, go to the System Overview dashboard and double-click a host name.

Figure 58: *Health monitoring stats of ubuntu 22.04*

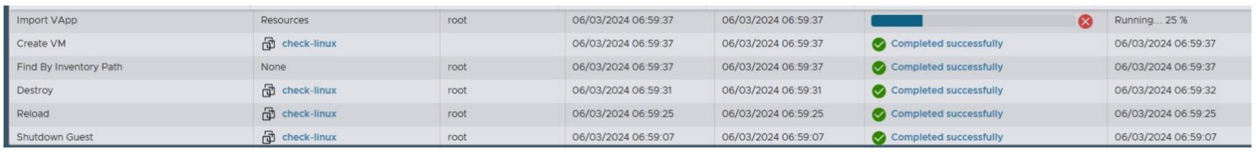
Figure 59 illustrates the Terraform script of snapshot restoration that will automatically delete the compromised machine and create new machine using the snapshot image of ubuntu 22.04.



```python
def snapshot_terraform_commands(directory):
    try:

        # Initialize Terraform
        subprocess.run(['terraform', 'init'], check=True, cwd=directory)

        # Generate execution plan
        subprocess.run(['terraform', 'plan', '-input=false'], check=True, cwd=directory)

        subprocess.run(['terraform', 'destroy', '-auto-approve'], check=True, cwd=directory)

        # Apply changes
        subprocess.run(['terraform', 'apply', '-auto-approve'], check=True, cwd=directory)

        return "Terraform commands executed successfully"
    except subprocess.CalledProcessError as e:
        raise Exception(f"Failed to execute Terraform commands: {e}")
```

Figure 59: *Terraform script of snapshot restoration*

Figure 60 illustrates the snapshot restoration of compromised machine ubuntu 22.04



Import VApp	Resources	root	06/03/2024 06:59:37	06/03/2024 06:59:37		Running... 25 %
Create VM	check-linux		06/03/2024 06:59:37	06/03/2024 06:59:37	✓ Completed successfully	06/03/2024 06:59:37
Find By Inventory Path	None	root	06/03/2024 06:59:37	06/03/2024 06:59:37	✓ Completed successfully	06/03/2024 06:59:37
Destroy	check-linux	root	06/03/2024 06:59:31	06/03/2024 06:59:31	✓ Completed successfully	06/03/2024 06:59:32
Reload	check-linux	root	06/03/2024 06:59:25	06/03/2024 06:59:25	✓ Completed successfully	06/03/2024 06:59:25
Shutdown Guest	check-linux	root	06/03/2024 06:59:07	06/03/2024 06:59:07	✓ Completed successfully	06/03/2024 06:59:07

Figure 60: *Snapshot restoration of Ubuntu 22.04*

After terraform snapshot restoration scripts run first it will destroy the compromised honeypot then automatically create the new ubuntu 22.04 machine using saved snapshot image.

Chapter 7

Conclusion

7. Conclusion

The objectives that were mentioned in chapter 1 have been achieved successfully. A robust control management and health monitoring system was developed, capable of gathering threat intelligence, automated deployment of honeypots and machine respawning.

Through automated deployment, organizations can strategically place decoy systems to lure potential attackers away from critical assets, enabling the collection of valuable intelligence on emerging threats. Furthermore, the integration of health monitoring ensures the continuous effectiveness of honeypots by promptly detecting any irregularities or potential issues. This vigilance not only enhances threat detection capabilities but also supports the ongoing improvement of cybersecurity defenses.

The inclusion of signature-based threat detection and mapping of honeypot logs to the MITRE ATT&CK framework further strengthens the project's ability to identify and categorize cyber threats. By leveraging predefined patterns of known threats and providing a structured framework for threat analysis, organizations can enhance their overall cybersecurity posture and better understand the tactics employed by malicious actors.

The machine re-spawning functionality ensures the resilience of the deception platform by restoring compromised or malfunctioning honeypots in accordance with health monitoring guidelines. In essence, "Hive Sentinel: The Hacker's Confine" represents a crucial step forward in cybersecurity, empowering organizations to stay ahead of cyber adversaries and fortify their defenses in an ever-evolving threat landscape.

7.1 Future Work:

1. **Integration with Cuckoo Sandbox:**

   Incorporate Cuckoo Sandbox with the existing virtual machines to enhance malware analysis capabilities. This will allow for automated dynamic analysis of suspicious files and URLs within the honeypot environment, providing deeper insights into the behavior and impact of potential threats.

2. **Advanced Machine Learning Algorithms:**

Implement advanced machine learning algorithms to improve threat detection and prediction. These algorithms can analyze patterns and anomalies in real-time, enhancing the system's ability to identify and respond to new and evolving threats more effectively.

3. **Expanded Threat Intelligence Sources:**

   Integrate additional threat intelligence sources such as VirusTotal, AlienVault, or IBM X-Force Exchange to enrich the data collected by the system. By leveraging a broader range of intelligence feeds, the system can stay updated on the latest threat trends and tactics, improving its overall effectiveness.

4. **Enhanced Incident Response Automation:**

   Enhance the system's incident response capabilities by automating more aspects of threat mitigation. This can include automated isolation of compromised systems, real-time alerts to security personnel, and predefined response actions to contain and neutralize threats quickly.

5. **Scalability Improvements:**

   Focus on improving the scalability of the system to support larger networks and more complex environments. This includes optimizing resource management and ensuring the system can handle increased loads without compromising performance.

6. **Integration with Existing Security Tools:**

   Enable seamless integration with existing security tools and platforms, such as SIEM (Security Information and Event Management) systems like Splunk or ArcSight, firewalls, and intrusion detection systems like Snort or Suricata. This will allow organizations to incorporate the honeypot system into their broader cybersecurity infrastructure more effectively.

By pursuing these future enhancements, the project can continue to evolve and provide even more robust protection against cyber threats, helping organizations maintain a strong security posture in an increasingly complex threat landscape.

References

References

[1] Team Acalvio. (2017). DECEPTION 1.0: HONEYPOTS ARE DEAD! In *Acalvio Deception Cyberdefense Manual* (pp. 8-15). Acalvio Technologies.

[2] Team Acalvio. (2017). DECEPTION 2.0: LONG LIVE HONEYPOTS! In *Acalvio Deception Cyberdefense Manual* (pp. 24-33). Acalvio Technologies.

[3] Lynda, boukela & Zhang, Gongxuan & Bouzefrane, Samia & Zhou, Junlong. (2020). An outlier ensemble for unsupervised anomaly detection in honeypots data. Intelligent Data Analysis. 24. 743-758. 10.3233/IDA-194656.

[4] Gowri, M. and Paramasivan, B. (2016), Rule-Based Anomaly Detection Technique Using Roaming Honeypots for Wireless Sensor Networks. ETRI Journal, 38: 1145-1152.

[5] Mendoza, M. A., & Amistadi, H. R. (2018). Machine learning for anomaly detection on VM and host performance metrics. MITRE CORP BEDFORD MA.

[6] Peter, E. Schiller, T., (2008) A Practical Guide to Honeypots, Washington University in St Louis, Missouri.

[7] M. Shukla and P. Verma, (2015), Honeypot: Concepts, Types, and Working, Department of Computer Engineering, SOCET, Ahmedabad, India.

[8] Mokue and M. Adams, (2007). Honeypots: Concepts, Approaches, and Challenges, Atlantic State University, Savannah, Georgia.

[9] Shahrivartehrani, S., & Abidin, S. A. Z. (2016). Dionaea Honeypot Implementation and Malware Analysis in Cloud Environment.

[10] Moore, C., & Al-Nemrat, A. (2015, September). An analysis of honeypot programs and the attack data collected. In International Conference on Global Security, Safety, and Sustainability (pp. 228-238). Springer, Cham.

[11] Koltys, K., 2013. An in-depth look at Kippo: an integration perspective, NASK, [Online].

[12] Strom, B. E., Battaglia, J. A., Kemmerer, M. S., Kupersanin, W., Miller, D. P., Wampler, C., ... & Wolf, R. D. (2017). Finding Cyber Threats with ATT and CK (registered trademark)-Based Analytics. MITRE CORP ANNAPOLIS JUNCTION MD.

[13] Daszczyszak, R., Ellis, D., Luke, S., & Whitley, S. (2019). TTP-Based Hunting. MITRE CORP MCLEAN VA.

[14] Pennington, A., Applebaum, A., Nickels, K., Schulz, T., Strom, B., & Wunder, J. (2019). Getting Started With ATT and CK. MITRE CORP MCLEAN VA.