TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Kristo Tammsoo 192928IVSB

Improving SOC Metrics Using Deception Technology

Bachelor's thesis

Supervisor: Mohammad Tariq Meeran
PhD in ICT

Madis Männik
MSc in Cyber Security

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kristo Tammsoo 192928IVSB

SOC mõõdikute parendamine pettetehnoloogia abil

Bakalaureusetöö

Juhendaja:	Mohammad Tariq Meeran PhD in ICT
	Madis Männik MSc in Cyber Security

Tallinn 2023

Author’s declaration of originality

I have duly acknowledged all the sources from which the ideas and extracts have been taken. The project is free from any plagiarism and has not been submitted elsewhere for publication.

Author: Kristo Tammsoo

14.05.2023

Abstract

This research work explores the integration of deception technology into Security Operations Centers (SOCs) to improve SOC metrics, focusing on the reconnaissance phase of the cyber kill chain model. Deception technology involves creating decoy assets to mislead attackers and gather valuable insights. The paper addresses challenges in using real security incidents to improve SOC metrics and highlights the benefits of deception technology.

The effectiveness of the prototype SOC system, incorporating deception technology, was evaluated, demonstrating improved accuracy in SOC metrics related to reconnaissance.

This paper contributes to the field by showcasing the potential of deception technology in enhancing SOC metrics and strengthening cybersecurity defenses. By leveraging deception technology, organizations can proactively detect and respond to threats, improving their overall cybersecurity posture.

This paper is written in English and is 39 pages long including 7 chapters, 1 table and 16 figures.

List of abbreviations

OSSIM	Open-Source Security Information and Event Management System
SIEM	Security Information and Event Management
SOC	Security Operations Center
SSH	Secure Shell

List of figures

1 Introduction

As cyber threats continue to evolve and become more sophisticated, it is essential for organizations to have an efficient and reliable way of measuring the effectiveness of their security operations. SOC metrics are crucial in responding to major security incidents that can impact critical infrastructures. These incidents can have far-reaching consequences and require a swift and effective response to minimize damage and mitigate risk. Deception technology can be used to measure and improve SOC metrics by providing more accurate and complete data on security incidents. Deception technology involves setting up fake assets, such as decoy services, to attract and deceive attackers. By monitoring these decoys, security teams can gain valuable insights into the tactics and techniques used by attackers.

Cyber attackers are finding new zero-day vulnerabilities at an alarming rate, which can lead to unauthorized access into a company's internal infrastructure. It's important to implement effective measures to prevent these attackers from causing harm. In the last two years, the number of zero-day vulnerabilities discovered exceeded the total number found in the previous decade [1].

Given the rapidly evolving threat landscape and the prevalence of zero-day vulnerabilities, the importance of improving SOC metrics cannot be overstated. It enables organizations to proactively defend against cyber threats, strengthen their security posture, and safeguard their valuable data and resources.

1.1 Problem statement

SOC metrics are crucial in evaluating an organization's security posture and measuring the effectiveness of security operations. They can aid in detecting incidents and responding to them quickly. However, challenges arise when relying on real incidents to improve metrics as the accuracy and completeness of data may be compromised. As cyber attackers continue to find new zero-day exploits, organizations are increasingly vulnerable to unauthorized access to their internal infrastructure. The challenge is to implement effective measures to prevent attackers from exploiting these vulnerabilities and causing harm.

1.2 Research questions

What are the challenges in using real security incidents to improve SOC metrics?

How can organizations improve the accuracy and completeness of data used to measure SOC metrics?

What are the key considerations when implementing deception technology?

1.3 Research goal

To develop guidelines for effectively integrating deception technology into an organization's existing security infrastructure and evaluate the effectiveness of deception technology in improving SOC metrics.

2 Background

The increasing sophistication of cyber threats poses a serious challenge to organizations' security operations. As attackers continue to find new vulnerabilities, organizations need to have effective and reliable ways of measuring the effectiveness of their security operations. SOC metrics are essential in the timely and efficient response to significant security incidents, which can have wide-ranging impacts and necessitate quick and effective actions to minimize harm and manage risk effectively. However, relying solely on real security incidents to improve SOC metrics can be challenging, as the accuracy and completeness of data may be compromised.

To address this challenge, organizations are turning to deception technology, which involves setting up fake assets to attract and deceive attackers. By monitoring these decoys, security teams can gain valuable insights into the tactics and techniques used by attackers, improving the accuracy and completeness of data used to measure SOC metrics. However, the effective use of deception technology requires careful consideration of several key factors.

In this section, we will explore the existing research related to deception technology and its usage to improve SOC metrics. We will examine the challenges associated with using real incidents to improve SOC metrics and the potential benefits of using deception technology. Additionally, we will identify key considerations when implementing deception technology, such as the selection of appropriate decoys and the integration of deception technology into an organization's existing security infrastructure.

Additionally, an assessment of how effective deception technology is in improving SOC metrics based on existing research in this domain.

2.1 SOC metrics importance

The importance of measuring the effectiveness of security operations centers (SOCs) has been recognized as critical in responding to cyber threats that continue to evolve and become more sophisticated. According to the Ponemon Institute, SOC metrics are essential in evaluating an organization's security posture and measuring the effectiveness of security operations [2]. The study found that organizations that measure SOC effectiveness have a better understanding of their security risks and are better equipped to respond to security incidents quickly.

Furthermore, Abbas Ahmed's study on security metrics and risks emphasized the importance of SOC metrics in detecting security incidents and assessing risk [3]. The study found that effective SOC metrics provide insights into the effectiveness of an organization's security posture and can aid in detecting incidents and responding to them quickly. In addition, the study highlighted the importance of having accurate and complete data to measure SOC metrics, as the quality of the data can affect the effectiveness of the SOC.

Both studies demonstrate the importance of SOC metrics in evaluating an organization's security posture and measuring the effectiveness of security operations. They also emphasize the need for accurate and complete data to measure SOC metrics, which can aid in detecting incidents and responding to them quickly. Overall, these studies highlight the critical role that SOC metrics play in responding to cyber threats, minimizing damage, and mitigating risk.

2.2 Measuring SOC metrics

Measuring the effectiveness of SOCs is crucial in ensuring the security of an organization's assets. However, SOC metrics can be difficult to measure due to the lack of standardization and the complexity of SOC operations. A systematic review conducted by Agyepong et al. identified twelve challenges faced by SOC analysts, including the volume of alerts presented to the analyst, the number of false positive alerts, false negatives, sophisticated attacks, incident handling/management complexity, skills/experience shortage, inadequate communication between teams, tacit knowledge, manual and repetitive processes, workloads, analysts burnout, and the lack of adequate metrics and measures for assessing the efforts of analysts. However, the study also identified existing metrics and measures for assessing the performance of analysts, including time to detect an incident, average time to detect an incident, average time taken to respond to an incident, number of alerts analyzed/unanalyzed by an analyst at the end of a shift, number of tickets closed per day, number of incidents detected within a specific timeframe, time spent on operations by the analyst, time spent on each ticket, a measure of the competency and experience of the analysts, success stories, the quality of incident reports, and the quality of analysis. Despite this progress, further research is needed to improve existing measures and metrics for assessing SOC analyst performance [4].

2.3 Deception Technologies

Deception technology has gained significant attention in recent years as a proactive approach to improve an organization's security posture. The technique involves creating decoy or trap resources that mimic legitimate servers, applications, and data. One of its primary benefits is decreasing an attacker's dwell time on the network. By using decoy assets that appear legitimate, deception technology can make it difficult for cybercriminals to steal an organization's actual assets. IT teams can focus their efforts on studying the behaviour and movements of attackers, which helps expedite the average time to detect and remediate threats. Additionally, deception technology helps reduce alert fatigue by providing additional alerts that help IT understand malicious behaviour and track the activities of attackers.

However, there are risks associated with deception technology. Cyber criminals have escalated the size, scope, and sophistication of their attacks, and a breach may be greater than what the deception server and its associated shadow or mock assets can handle. Additionally, cyber criminals may quickly determine that they are being tricked, causing them to abort the attack and return even stronger. Therefore, to function properly, deception technology must not be obvious to an enterprise's employees, contractors, or customers. Deception technology is important because it decreases the attacker's dwell time on the network, expedites the average time to detect and remediate threats, and reduces alert fatigue. By diverting attackers to decoy resources, organizations can focus their efforts on studying their behaviors and movements and quickly detect and address threats [5].

The Cyber Kill Chain model describes the stages of a cyber-attack, from initial reconnaissance to data exfiltration [6]. This model can be used to develop effective countermeasures against cyber-attacks. To expand on the concept of deception technology in the context of the cyber kill chain, a study conducted by Almeshekah and Spafford proposed a table that maps the use of deception techniques to each phase of the cyber kill chain [7]. Specifically, Table 1 below highlights the various deception techniques that can be utilized in the reconnaissance, weaponization and delivery, exploitation and installation, command and control, lateral movement and persistence, and staging and exfiltration phases of the cyber kill chain.

In the reconnaissance phase, deception techniques such as artificial ports and fake sites can be used to mislead attackers and lure them into traps. In the weaponization and delivery phase, creating artificial bouncing back and sticky honeypots can help to prevent attackers from delivering their malicious payloads successfully. Similarly, in the exploitation and installation phase, creating artificial exploitation responses can prevent attackers from successfully exploiting vulnerabilities.

In the command-and-control phase, honeypots can be used to mislead attackers into believing that they have gained control over a system, while in the lateral movement and persistence phase, honey accounts and honeyfiles can be used to gather information about the attacker and their tactics. Finally, in the staging and exfiltration phase, honeytokens, endless files, and fake keys can be used to create fake data that attackers can waste time trying to exfiltrate. [7]

Table 1. Mapping deception to the kill-chain model [7]

Cyber kill-chain phase	Deception
Reconnaissance	Artificial ports, fake sites
Weaponization and delivery	Create artificial bouncing back, sticky honeypots
Exploitation and installation	Create artificial exploitation response
Command and control (operation)	Honeypot
Lateral movement and persistence	HoneyAccounts, HoneyFiles
Staging and exfiltration	Honeytokens, endless files, fake keys

3 Methodology

This section outlines the methodology used to design and develop a prototype SOC system that leverages deception technology to provide accurate metrics, with a specific focus on the reconnaissance phase of the cyber kill chain model.

The goal of this research is to create a proof-of-concept that demonstrates how deception technology can be integrated into a SOC system to improve the accuracy of security metrics, specifically in identifying and mitigating reconnaissance attempts. The methodology begins with a literature review of current research and industry practices related to SOC metrics and deception technology, with a particular focus on how deception technology can be applied to the reconnaissance phase. The next step is prototyping, which involves the creation of a working prototype system that incorporates deception technology, with an emphasis on developing methods for detecting and responding to reconnaissance attempts. The prototype system is designed to collect and analyse data in real-time, providing accurate and timely metrics to evaluate the effectiveness of the SOC system in detecting reconnaissance attempts.

Finally, data collection and analysis are conducted to evaluate the effectiveness of the prototype system, specifically in terms of how well it can detect and respond to reconnaissance attempts. This involves the collection of data from the prototype system, including metrics related to SOC effectiveness and incident response, with a focus on the accuracy of the metrics related to the reconnaissance phase of the cyber kill chain model. The data is then analysed to determine the impact of deception technology on the accuracy of SOC metrics for reconnaissance attempts.

3.1 Review and analysis of existing research in this domain

Several studies have evaluated the effectiveness of deception technology in detecting and responding to cyber threats, and its potential for reducing the attack surface of organizations. For instance, the Tularosa study conducted by Kimberly Ferguson-Walter, Temmie Shade, and Andrew Rogers aimed to quantify the efficacy of cyber deception by examining how attackers behave when they encounter deception technology [8]. The data that was collected from the original Tularosa study was used to conduct a thorough analysis by Kimberly J. Ferguson-Walter, et al. which evaluated the effectiveness of both decoy-based and psychological cyber deception. The study found that both decoy-based and psychological deception were effective in deterring attacks. However, decoy-based deception was more effective at detecting and deterring attacks than psychological deception. The researchers also found that the effectiveness of decoy-based deception was dependent on the number and diversity of decoys used. [9]

3.2 Prototyping

The prototyping phase involves the creation of a working proof-of-concept system that incorporates deception technology to improve the accuracy of SOC metrics. The prototype system is designed to collect and analyze data in real-time, providing accurate and timely metrics to evaluate the effectiveness of the SOC system.

The prototyping phase consists of three main stages: planning, development, and testing. During the planning stage, the requirements and objectives for the prototype system are defined, and the necessary hardware and software components are identified. The development stage involves the actual creation of the prototype system, including the integration of deception technology components such as artificial ports, fake sites, and fake services. The testing stage involves the evaluation of the prototype system's performance, including its ability to collect and analyze data in real-time, its accuracy in detecting and responding to security incidents, and its overall effectiveness in improving SOC metrics.

The prototype system will be developed using open-source software and tools, allowing for easy customization and modification as needed. The system will also be designed to integrate with existing SOC systems and workflows, making it easy to deploy and use in a production environment.

3.3 Data collection and analysis method

To assess the effectiveness of the prototype SOC system, an experimental setup was created consisting of one web server running on Apache web server software. Deception technology was implemented using cowrie, as well as other deception techniques such as fake directory structures. Rsyslog was used for collecting logs and Logstash for forwarding system logs to the SIEM. The setup was designed to closely mimic a real-world environment, with the web server exposed to the Internet and subject to attacks from various sources.

By using this experimental setup, data was gathered on the effectiveness of the prototype SOC system and the results obtained were validated. The use of realistic scenarios and environments provides a more accurate representation of the system's performance and allows for better evaluation of its effectiveness.

Data collection and analysis were conducted to evaluate the effectiveness of the prototype system, specifically in terms of how well it can detect and respond to reconnaissance attempts. This involved the collection of data from the prototype system, including metrics related to SOC effectiveness and incident response, with a focus on the accuracy of the metrics related to the reconnaissance phase of the cyber kill chain model.

4 Experimental setup

To assess the effectiveness of the prototype SOC system, an experimental setup was created consisting of one web server running on Apache web server software, a cowrie honeypot, and an OpenSearch cluster. Deception technology was implemented using Cowrie, as well as other deception techniques such as fake directory structures. Rsyslog was used for collecting logs and Logstash for forwarding system logs to the centralized logging server.

The setup was designed to closely mimic a real-world environment, with the web servers exposed to the Internet and subject to attacks from various sources.

By using this experimental setup, data was gathered on the effectiveness of the prototype SOC system and the results obtained were validated. The use of realistic scenarios and environments provides a more accurate representation of the system's performance and allows for better evaluation of its effectiveness.

4.1 Prototyping

The following experimental setup was used:

- Hardware: Dell PowerEdge R430 server with 1x Intel Xeon E5-2620 CPU, 64GB RAM, and 4x 500GB HDDs in RAID 10 configuration
- Operating system: Ubuntu Server 18.04 LTS, Ubuntu Server 20.04 LTS
- Network topology: Apache2 web server, monitoring server
- Log management and correlation tool: Opensearch
- System Logging: Rsyslog for collecting and Logstash for forwarding system logs to the SIEM
- Data Source: Cowrie, apache2 access logs
- Realistic Scenario: A test environment that closely mimics a real-world environment, with the web servers exposed to the Internet and subject to attacks from various sources.

A fresh Ubuntu Server 20.04 LTS virtual machine was created in the same subnet as the web server for the prototype. The virtual machine was configured with open-source SIEM software. Two open-source SIEM options were evaluated for the software: OSSIM and OpenSearch. The first option was OSSIM, which is developed by AT&T as a fully open source SIEM under the GNU General Public License [10]. The second option was OpenSearch, a community driven, open-source fork of Elasticsearch and Kibana [11].

When selecting the appropriate open-source SIEM software to use for the prototype's Ubuntu Server 20.04 LTS virtual machine, two key factors were considered: the latest release date and ease of installation. OSSIM, which is developed by AT&T as a fully open-source SIEM under the GNU General Public License, had its latest stable release in May 2022. In contrast, OpenSearch, a community-driven, open-source fork of Elasticsearch and Kibana, was found to be currently being actively developed, with the latest stable release being in March of 2023. Another factor that was considered was that OSSIM required an entire virtual machine to run the software, while OpenSearch could be directly installed onto a docker instance in a Debian-based Linux virtual machine. Based on these factors, the decision was made to use OpenSearch as the SIEM for the prototype.

To install and configure the OpenSearch software for the prototype, the official documentation for OpenSearch was utilized. The documentation provided instructions and guidance on how to properly install and configure the SIEM software on the virtual machine. As a result, a single node OpenSearch cluster was configured, the docker-compose.yml file can be found in the appendix 1.

4.2 Use of Deception Technology to collect data

The diagram of how deception technology was implemented to ingest data into the OpenSearch cluster is presented in Figure 1.

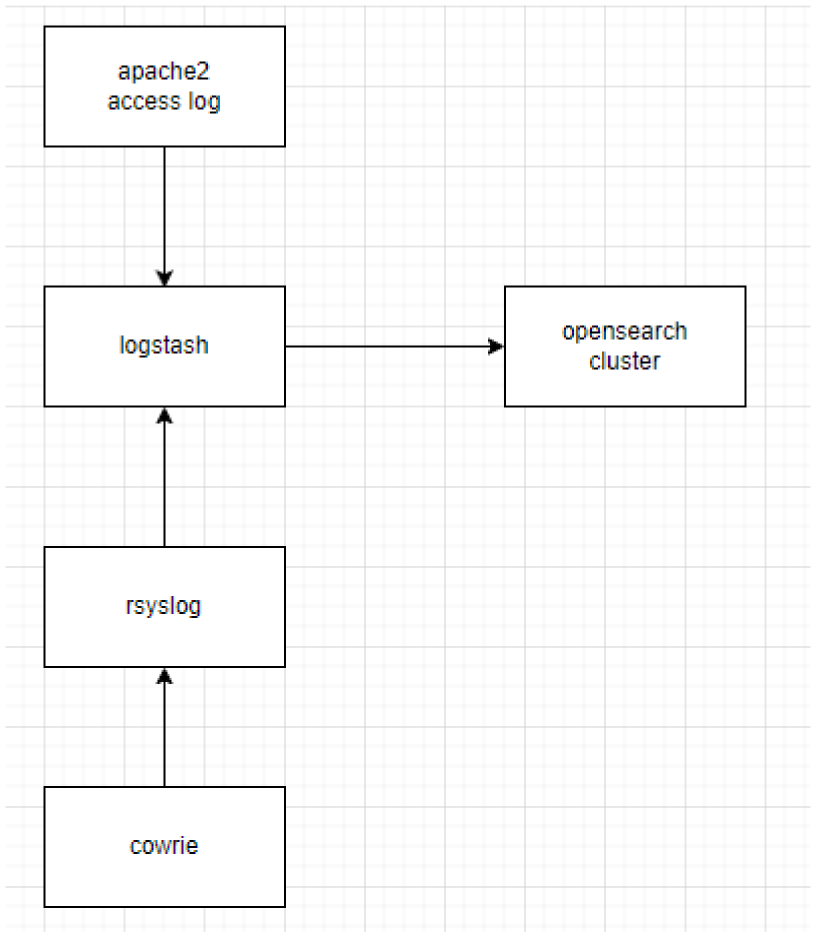


Figure 1. Diagram of log ingestion from deception technology

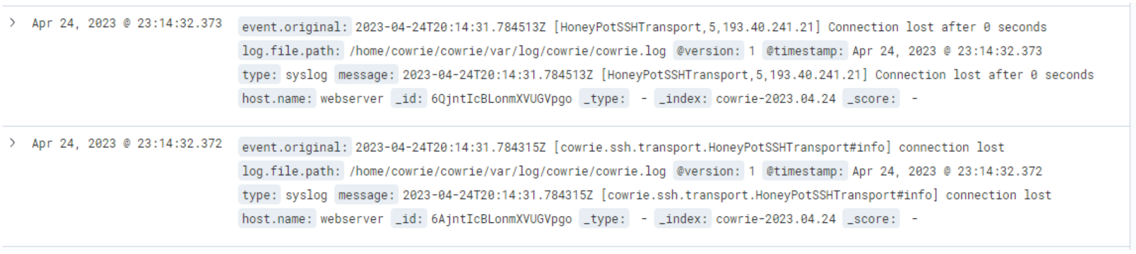
To achieve the above result a private virtual switch was created for the virtual machines and the machines were configured with a private IP address. The data from Apache2 access logs and syslog, was sent to the Opensearch cluster via the private IP addresses assigned to the virtual machines.

4.3 Improving SOC metrics with data from deception technology

One method for improving SOC metrics through deception technology involves the use of open-source software such as Cowrie to open ports on a machine and log interactions with those ports to syslog [12]. By using deception technology to monitor interactions with the fake ports created by the open-source software, it is possible to detect certain types of cyber-attacks, such as port scanning or reconnaissance attacks. This can provide valuable insights into the behaviour of attackers and help SOC teams improve their overall security posture. Additionally, by analysing the data collected from these interactions, it may be possible to identify patterns of behaviour that could indicate a potential attack in progress or allow for early detection of a new threat.

Cowrie was installed and configured on the web server by following the official documentation provided by the developers. The configuration file that was used can be found in the appendix of this paper.

When executing a nmap port scan against the webserver logs are generated by Cowrie and are sent to Opensearch, as shown in Figure 2.



```
> Apr 24, 2023 @ 23:14:32.373  event.original: 2023-04-24T20:14:31.784513Z [HoneyPotSSHTransport,5,193.40.241.21] Connection lost after 0 seconds
                               log.file.path: /home/cowrie/cowrie/var/log/cowrie/cowrie.log @version: 1 @timestamp: Apr 24, 2023 @ 23:14:32.373
                               type: syslog message: 2023-04-24T20:14:31.784513Z [HoneyPotSSHTransport,5,193.40.241.21] Connection lost after 0 seconds
                               host.name: webserver _id: 6QjntIcBLonmXVUGVpgo _type: - _index: cowrie-2023.04.24 _score: -

> Apr 24, 2023 @ 23:14:32.372  event.original: 2023-04-24T20:14:31.784315Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
                               log.file.path: /home/cowrie/cowrie/var/log/cowrie/cowrie.log @version: 1 @timestamp: Apr 24, 2023 @ 23:14:32.372
                               type: syslog message: 2023-04-24T20:14:31.784315Z [cowrie.ssh.transport.HoneyPotSSHTransport#info] connection lost
                               host.name: webserver _id: 6AjntIcBLonmXVUGVpgo _type: - _index: cowrie-2023.04.24 _score: -
```

Figure 2. Opensearch visualization of the received logs

Interactions with the port exposed by Cowrie can serve as an early warning sign of a potential cyber-attack, as Cowrie actively detects and captures all interactions. With this information, SOC teams can take proactive measures to mitigate the threat and prevent further exploitation of the network. By leveraging the power of deception technology and log analysis, it is possible to improve SOC metrics and enhance the overall security posture of an organization.

The Cowrie honeypot was left running on emulated shell mode for a total of 4 days, during which 10,822 hits were recorded by the Opensearch cluster, see Figure 3:

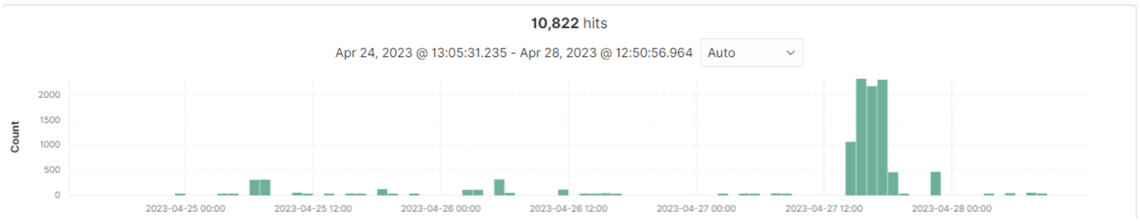
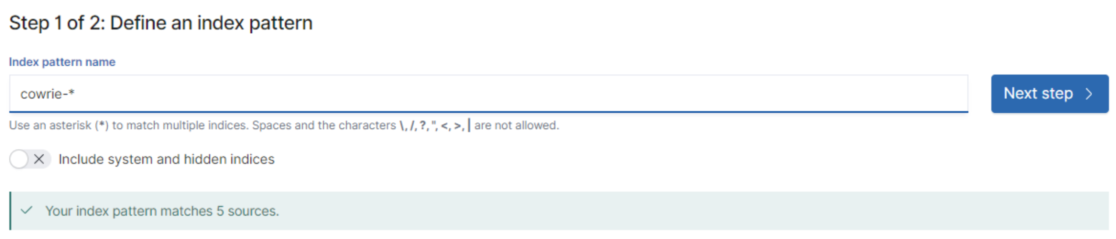


Figure 3. Hits recorded by the OpenSearch cluster from Cowrie logs.

Another method to deploy deception technology involves creating fake directories on a web server that can be triggered by scanning tools such as DirBuster. Apache2 access logs can capture attacker behavior, enabling identification of attack vectors and suspicious activity. By integrating this data with OpenSearch and using detectors to identify patterns of suspicious behavior, SOC metrics can be enhanced in terms of both accuracy and efficiency.

Two index patterns (cowrie-* and apache-access-*) were created to retrieve the data from Logstash, see Figure 4 and 5:



Step 1 of 2: Define an index pattern

Index pattern name

cowrie-*

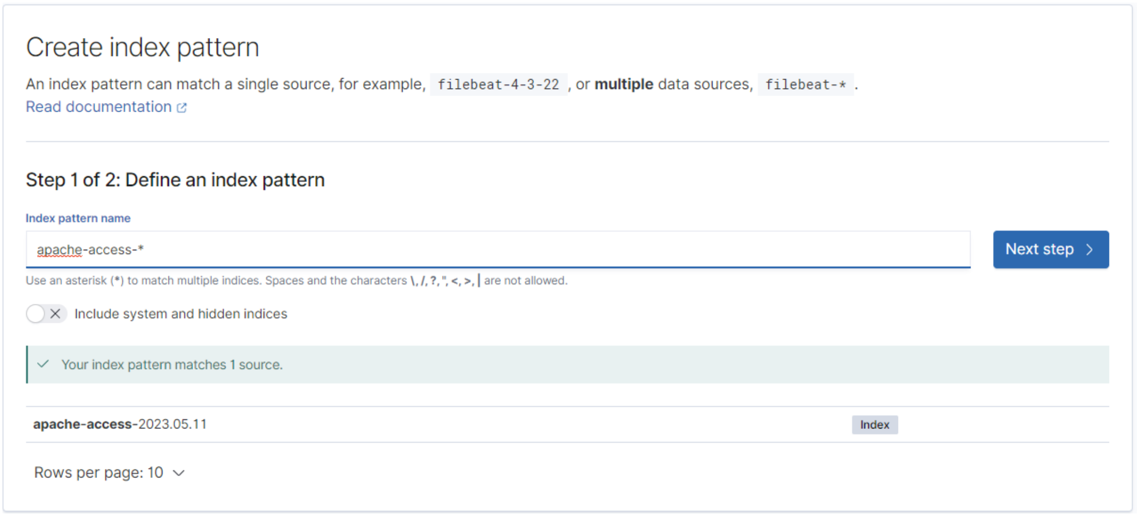
Next step

Use an asterisk (*) to match multiple indices. Spaces and the characters \, /, ?, ", <, >, | are not allowed.

Include system and hidden indices

Your index pattern matches 5 sources.

Figure 4. Creating an index pattern for Cowrie logs.



Create index pattern

An index pattern can match a single source, for example, `filebeat-4-3-22`, or **multiple** data sources, `filebeat-*`.

Read documentation

Step 1 of 2: Define an index pattern

Index pattern name

apache-access-*

Next step

Use an asterisk (*) to match multiple indices. Spaces and the characters \, /, ?, ", <, >, | are not allowed.

Include system and hidden indices

Your index pattern matches 1 source.

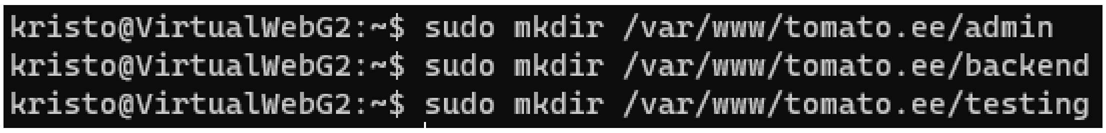
apache-access-2023.05.11 Index

Rows per page: 10

Figure 5. Creating an index pattern for Apache2 access logs.

As a proof-of-concept, an experiment was conducted where fake directories were created on a web server based on examples of directories that are typically scanned by the popular tool DirBuster. Examples of directories typically scanned by DirBuster were obtained from the official Kali Linux Packages GitLab repository [13].

Figure 6 illustrates the creation of fake directories on the web server:



```
kristo@VirtualWebG2:~$ sudo mkdir /var/www/tomato.ee/admin
kristo@VirtualWebG2:~$ sudo mkdir /var/www/tomato.ee/backend
kristo@VirtualWebG2:~$ sudo mkdir /var/www/tomato.ee/testing
```

Figure 6. Creating fake directories on the web server

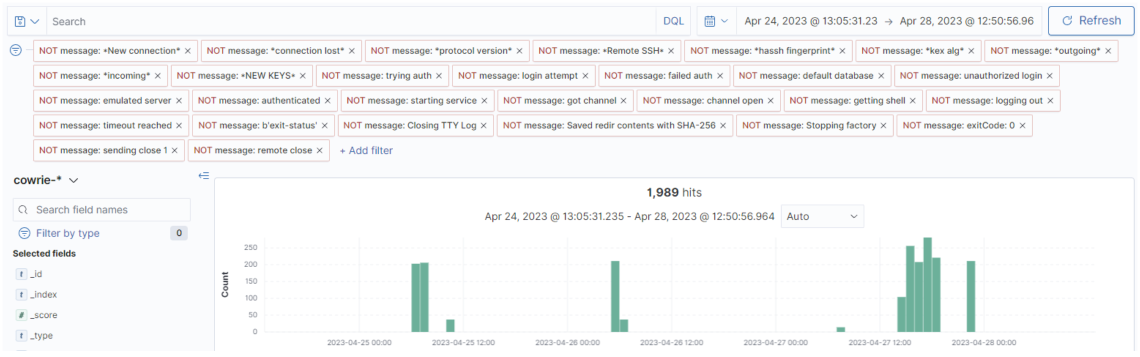
Interactions with the fake directories on the website are logged in Apache access logs, providing insight into potential attacks and other suspicious behavior. Logstash can forward these logs to the OpenSearch cluster, where they can be analysed.

5 Result and analysis (discussion)

The experimental setup described in the previous section was used to evaluate the effectiveness of the prototype SOC system in detecting and responding to reconnaissance attempts. Data was collected and analysed to determine the accuracy of SOC metrics related to the reconnaissance phase of the cyber kill chain model.

This involved the creation of a web server and a monitoring server, as well as the deployment of the Opensearch cluster, Cowrie honeypot and fake directories to detect interactions with open ports.

To analyse the data generated by Cowrie, many filters were added to the timeframe of which the honeypot was running, which filtered out basic debug messages from all the hits. After applying the filters there were 1989 hits left, see Figure 7:



Search | DQL | Apr 24, 2023 @ 13:05:31.23 → Apr 28, 2023 @ 12:50:56.96 | Refresh

NOT message: *New connection* ×
NOT message: *connection lost* ×
NOT message: *protocol version* ×
NOT message: *Remote SSH* ×
NOT message: *hassh fingerprint* ×
NOT message: *kex alg* ×
NOT message: *outgoing* ×
NOT message: *incoming* ×
NOT message: *NEW KEYS* ×
NOT message: trying auth ×
NOT message: login attempt ×
NOT message: failed auth ×
NOT message: default database ×
NOT message: unauthorized login ×
NOT message: emulated server ×
NOT message: authenticated ×
NOT message: starting service ×
NOT message: got channel ×
NOT message: channel open ×
NOT message: getting shell ×
NOT message: logging out ×
NOT message: timeout reached ×
NOT message: b'exit-status' ×
NOT message: Closing TTY Log ×
NOT message: Saved redir contents with SHA-256 ×
NOT message: Stopping factory ×
NOT message: exitCode: 0 ×
NOT message: sending close 1 ×
NOT message: remote close ×
+ Add filter

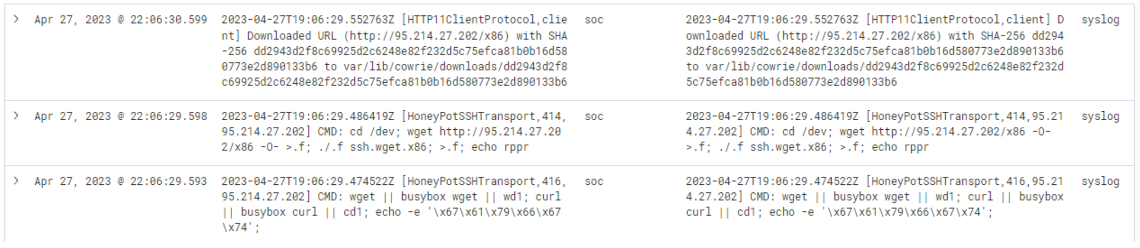
cowrie-*

1,989 hits

Apr 24, 2023 @ 13:05:31.235 - Apr 28, 2023 @ 12:50:56.964 Auto

Figure 7. Filtering the hits recorded from Cowrie honeypot.

Cowrie provides an interactive shell for the adversary and their actions within the shell can be monitored. Figure 8 shows an example of the logs that Cowrie generated that track the adversary's movement and actions within the shell:



```
>  Apr 27, 2023 @ 22:06:30.599  2023-04-27T19:06:29.552763Z [HTTP11ClientProtocol,clie   soc        2023-04-27T19:06:29.552763Z [HTTP11ClientProtocol,client] D   syslog
                                nt] Downloaded URL (http://95.214.27.202/x86) with SHA             ownloaded URL (http://95.214.27.202/x86) with SHA-256 dd294
                                -256 dd2943d2f8c69925d2c6248e82f232d5c75efca81b0b16d58             3d2f8c69925d2c6248e82f232d5c75efca81b0b16d580773e2d890133b6
                                0773e2d890133b6 to var/lib/cowrie/downloads/dd2943d2f8             to var/lib/cowrie/downloads/dd2943d2f8c69925d2c6248e82f232d
                                c69925d2c6248e82f232d5c75efca81b0b16d580773e2d890133b6             5c75efca81b0b16d580773e2d890133b6

>  Apr 27, 2023 @ 22:06:29.598  2023-04-27T19:06:29.486419Z [HoneyPotSSHTransport,414,   soc        2023-04-27T19:06:29.486419Z [HoneyPotSSHTransport,414,95.21   syslog
                                95.214.27.202] CMD: cd /dev; wget http://95.214.27.20              4.27.202] CMD: cd /dev; wget http://95.214.27.202/x86 -O-
                                2/x86 -O- >.f; ./.f ssh.wget.x86; >.f; echo rppr                   >.f; ./.f ssh.wget.x86; >.f; echo rppr

>  Apr 27, 2023 @ 22:06:29.593  2023-04-27T19:06:29.474522Z [HoneyPotSSHTransport,416,   soc        2023-04-27T19:06:29.474522Z [HoneyPotSSHTransport,416,95.21   syslog
                                95.214.27.202] CMD: wget || busybox wget || wd1; curl              4.27.202] CMD: wget || busybox wget || wd1; curl || busybox
                                || busybox curl || cd1; echo -e '\x67\x61\x79\x66\x67              curl || cd1; echo -e '\x67\x61\x79\x66\x67\x74';
                                \x74';
```

Figure 8. Example of logs generated by Cowrie.

The adversary that interacted with the shell executed the following series of commands on the system:

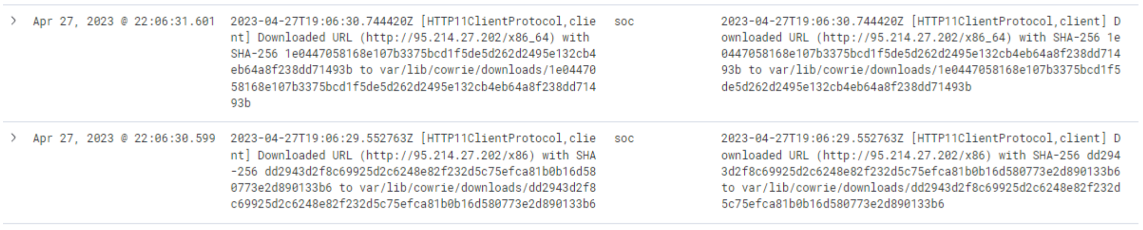
cd /dev; wget http://95.214.27.202/x86 **-O-** >.f; **./.f** ssh.wget.x86; >**.f; echo** rppr

These commands can be analysed:

1. **cd /dev**: Changes the working directory to **/dev**, which contains device files that are essential for the system to function properly.

2. **wget http://95.214.27.202/x86 -O- >.f**: Downloads the file from the specified URL (**http://95.214.27.202/x86**) using the **wget** command. The **-O-** flag specifies that the downloaded file should be sent to the standard output (instead of saving it to a file with the same name). The **> .f** part of the command redirects the standard output to a file named **.f** in the **/dev** directory.

3. **./.f ssh.wget.x86**: Executes the **.f** file that was just downloaded, passing **ssh.wget.x86** as an argument. From the logs it can be traced, that this command executed a wget command, which downloaded a file called x86_64, see Figure 9:



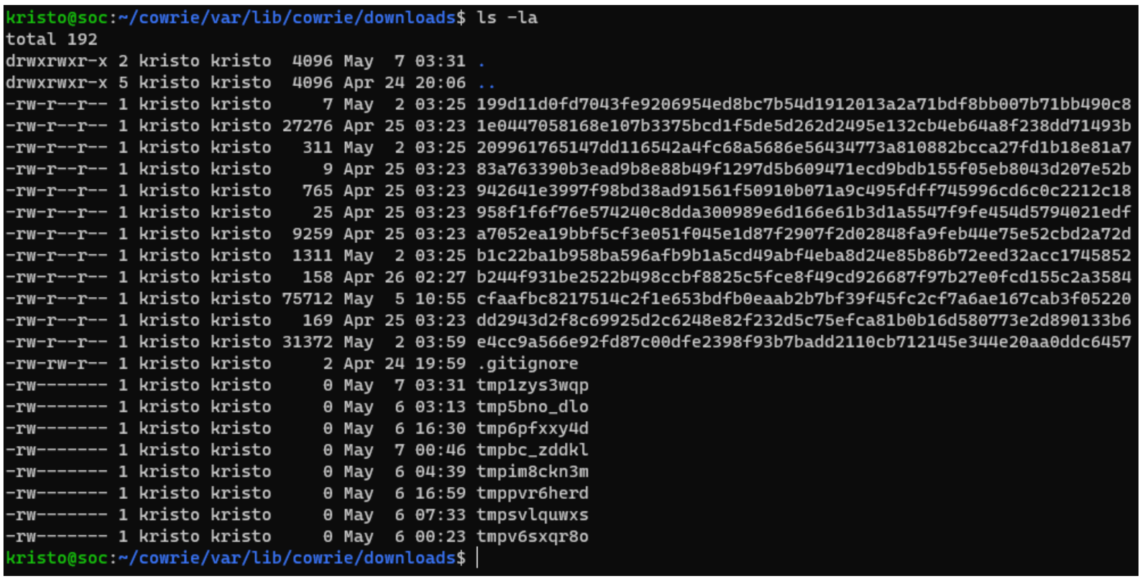
```
>  Apr 27, 2023 @ 22:06:31.601  2023-04-27T19:06:30.744420Z [HTTP11ClientProtocol,clie   soc   2023-04-27T19:06:30.744420Z [HTTP11ClientProtocol,client] D
                                nt] Downloaded URL (http://95.214.27.202/x86_64) with          ownloaded URL (http://95.214.27.202/x86_64) with SHA-256 1e
                                SHA-256 1e0447058168e107b3375bcd1f5de5d262d2495e132cb4         0447058168e107b3375bcd1f5de5d262d2495e132cb4eb64a8f238dd714
                                eb64a8f238dd71493b to var/lib/cowrie/downloads/1e04470         93b to var/lib/cowrie/downloads/1e0447058168e107b3375bcd1f5
                                58168e107b3375bcd1f5de5d262d2495e132cb4eb64a8f238dd714         de5d262d2495e132cb4eb64a8f238dd71493b
                                93b

>  Apr 27, 2023 @ 22:06:30.599  2023-04-27T19:06:29.552763Z [HTTP11ClientProtocol,clie   soc   2023-04-27T19:06:29.552763Z [HTTP11ClientProtocol,client] D
                                nt] Downloaded URL (http://95.214.27.202/x86) with SHA         ownloaded URL (http://95.214.27.202/x86) with SHA-256 dd294
                                -256 dd2943d2f8c69925d2c6248e82f232d5c75efca81b0b16d58         3d2f8c69925d2c6248e82f232d5c75efca81b0b16d580773e2d890133b6
                                0773e2d890133b6 to var/lib/cowrie/downloads/dd2943d2f8         to var/lib/cowrie/downloads/dd2943d2f8c69925d2c6248e82f232d
                                c69925d2c6248e82f232d5c75efca81b0b16d580773e2d890133b6         5c75efca81b0b16d580773e2d890133b6
```

Figure 9. Trace of download log from command execution

4. **>.f**: Creates (or truncates) an empty file named **.f**. If the file already exists, its content will be removed.

5. **echo rppr**: Prints the string "rppr" to the console. Although this string appears to be arbitrary, it could potentially serve as the attacker's unique signature.

The downloaded file can also be analysed, as Cowrie has built-in functionality, which gets the hash of the file and stores a copy of the original file. Figure 10 shows all the files that were gathered:



```
kristo@soc:~/cowrie/var/lib/cowrie/downloads$ ls -la
total 192
drwxrwxr-x 2 kristo kristo  4096 May  7 03:31 .
drwxrwxr-x 5 kristo kristo  4096 Apr 24 20:06 ..
-rw-r--r-- 1 kristo kristo     7 May  2 03:25 199d11d0fd7043fe9206954ed8bc7b54d1912013a2a71bdf8bb007b71bb490c8
-rw-r--r-- 1 kristo kristo 27276 Apr 25 03:23 1e0447058168e107b3375bcd1f5de5d262d2495e132cb4eb64a8f238dd71493b
-rw-r--r-- 1 kristo kristo   311 May  2 03:25 209961765147dd116542a4fc68a5686e56434773a810882bcca27fd1b18e81a7
-rw-r--r-- 1 kristo kristo     9 Apr 25 03:23 83a763390b3ead9b8e88b49f1297d5b609471ecd9bdb155f05eb8043d207e52b
-rw-r--r-- 1 kristo kristo   765 Apr 25 03:23 942641e3997f98bd38ad91561f50910b071a9c495fdff745996cd6c0c2212c18
-rw-r--r-- 1 kristo kristo    25 Apr 25 03:23 958f1f6f76e574240c8dda300989e6d166e61b3d1a5547f9fe454d5794021edf
-rw-r--r-- 1 kristo kristo  9259 Apr 25 03:23 a7052ea19bbf5cf3e051f045e1d87f2907f2d02848fa9feb44e75e52cbd2a72d
-rw-r--r-- 1 kristo kristo  1311 May  2 03:25 b1c22ba1b958ba596afb9b1a5cd49abf4eba8d24e85b86b72eed32acc1745852
-rw-r--r-- 1 kristo kristo   158 Apr 26 02:27 b244f931be2522b498ccbf8825c5fce8f49cd926687f97b27e0fcd155c2a3584
-rw-r--r-- 1 kristo kristo 75712 May  5 10:55 cfaafbc8217514c2f1e653bdfb0eaab2b7bf39f45fc2cf7a6ae167cab3f05220
-rw-r--r-- 1 kristo kristo   169 Apr 25 03:23 dd2943d2f8c69925d2c6248e82f232d5c75efca81b0b16d580773e2d890133b6
-rw-r--r-- 1 kristo kristo 31372 May  2 03:59 e4cc9a566e92fd87c00dfe2398f93b7badd2110cb712145e344e20aa0ddc6457
-rw-rw-r-- 1 kristo kristo     2 Apr 24 19:59 .gitignore
-rw------- 1 kristo kristo     0 May  7 03:31 tmp1zys3wqp
-rw------- 1 kristo kristo     0 May  6 03:13 tmp5bno_dlo
-rw------- 1 kristo kristo     0 May  6 16:30 tmp6pfxxy4d
-rw------- 1 kristo kristo     0 May  7 00:46 tmpbc_zddkl
-rw------- 1 kristo kristo     0 May  6 04:39 tmpim8ckn3m
-rw------- 1 kristo kristo     0 May  6 16:59 tmppvr6herd
-rw------- 1 kristo kristo     0 May  6 07:33 tmpsvlquwxs
-rw------- 1 kristo kristo     0 May  6 00:23 tmpv6sxqr8o
kristo@soc:~/cowrie/var/lib/cowrie/downloads$ 
```

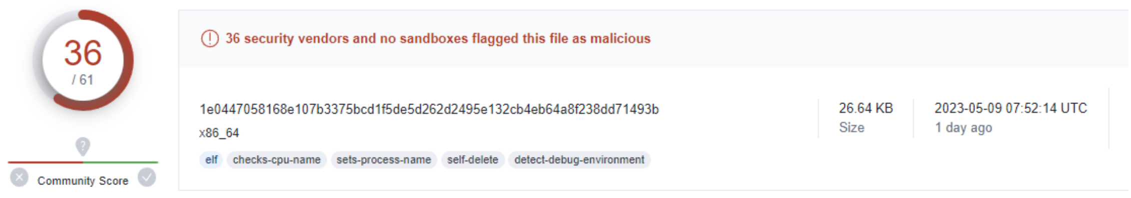
Figure 10. All downloads received from Cowrie.

From the log analysis above two files were found to be downloaded by the adversary with the file hashes of:

1. 1e0447058168e107b3375bcd1f5de5d262d2495e132cb4eb64a8f238dd71493b

2. dd2943d2f8c69925d2c6248e82f232d5c75efca81b0b16d580773e2d890133b6

The first hash was looked up in virustotal as on first glance the file appeared to be some kind of compiled binary. The virustotal search revealed the file to be malicious, see Figure 11:



36 / 61

Community Score

36 security vendors and no sandboxes flagged this file as malicious

1e0447058168e107b3375bcd1f5de5d262d2495e132cb4eb64a8f238dd71493b
x86_64

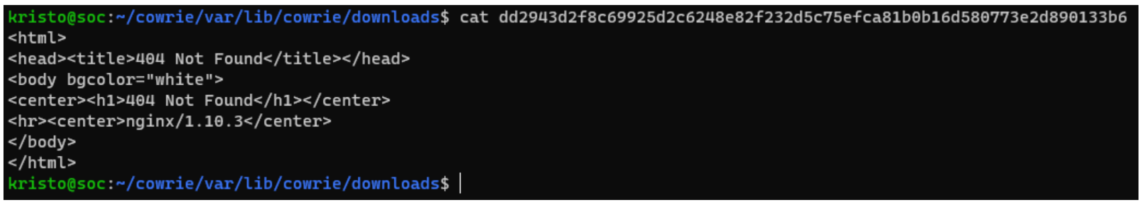
elf checks-cpu-name sets-process-name self-delete detect-debug-environment

26.64 KB
Size

2023-05-09 07:52:14 UTC
1 day ago

Figure 11. Virustotal result for the first hash.

The second hash was analysed manually as presented in Figure 12:

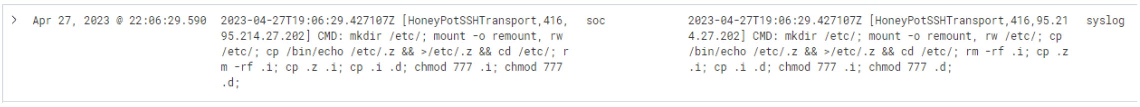


```
kristo@soc:~/cowrie/var/lib/cowrie/downloads$ cat dd2943d2f8c69925d2c6248e82f232d5c75efca81b0b16d580773e2d890133b6
<html>
<head><title>404 Not Found</title></head>
<body bgcolor="white">
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.10.3</center>
</body>
</html>
kristo@soc:~/cowrie/var/lib/cowrie/downloads$ 
```

Figure 12. Output of the second file in CLI

The contents of the file are the output of the second wget command that got executed. The content of the file is likely caused by the fact that the file did not exist on the adversary’s system and therefore the adversaries web server returned an error code.

Another example that shows the sophistication of the attacks that were attempted on the honeypot shell is presented in Figure 13:



```
>  Apr 27, 2023 @ 22:06:29.590  2023-04-27T19:06:29.427107Z [HoneyPotSSHTransport,416,   soc          2023-04-27T19:06:29.427107Z [HoneyPotSSHTransport,416,95.21   syslog
                                95.214.27.202] CMD: mkdir /etc/; mount -o remount, rw                4.27.202] CMD: mkdir /etc/; mount -o remount, rw /etc/; cp
                                /etc/; cp /bin/echo /etc/.z && >/etc/.z && cd /etc/; r               /bin/echo /etc/.z && >/etc/.z && cd /etc/; rm -rf .i; cp .z
                                m -rf .i; cp .z .i; cp .i .d; chmod 777 .i; chmod 777                .i; cp .i .d; chmod 777 .i; chmod 777 .d;
                                .d;
```

Figure 13. Example of attack sophistication

The commands that were executed are as follows:

1. **mount -o remount, rw /etc/**: This command remounts the **/etc** directory with read-write permissions. The **/etc** directory typically contains system configuration files, which are critical for the system's operation.

2. **cp /bin/echo /etc/.z && >/etc/.z**: This command copies the **/bin/echo** executable to **/etc/.z** and then truncates the file, effectively creating an empty file named **.z** in the **/etc** directory. The **&&** symbol ensures that the truncation is performed only if the copy operation is successful.

3. **cd /etc/**: Changes the working directory to **/etc**.

4. **rm -rf .i**: Recursively and forcefully removes the **.i** file or directory if it exists.

5. **cp .z .i**: Copies the **.z** file to **.i**.

6. **cp .i .d**: Copies the **.i** file to **.d**.

7. **chmod 777 .i; chmod 777 .d;**: Sets the permissions of both the **.i** and **.d** files to 777, which means read, write, and execute permissions for everyone.

The command sequence executed by the adversary within the Cowrie honeypot demonstrates a potential attempt to manipulate the system's configuration files and create files with insecure permissions. The adversary sets the permissions of the created files to 777, granting read, write, and execute permissions to everyone. This insecure configuration can be exploited by an attacker to further compromise the system, modify, or replace the content of these files, or use them as part of a larger attack. The data collected from the interactions with the emulated shell environment showcases the sophistication of attackers through their interactions with the deception technology.

The Cowrie honeypot in total successfully detected 430 unique connections, see Figure 14:

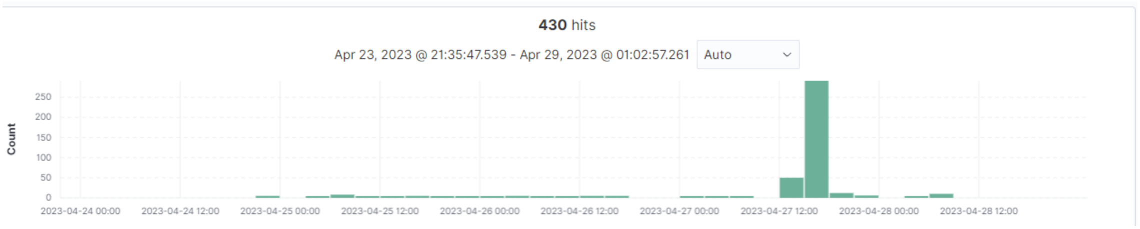
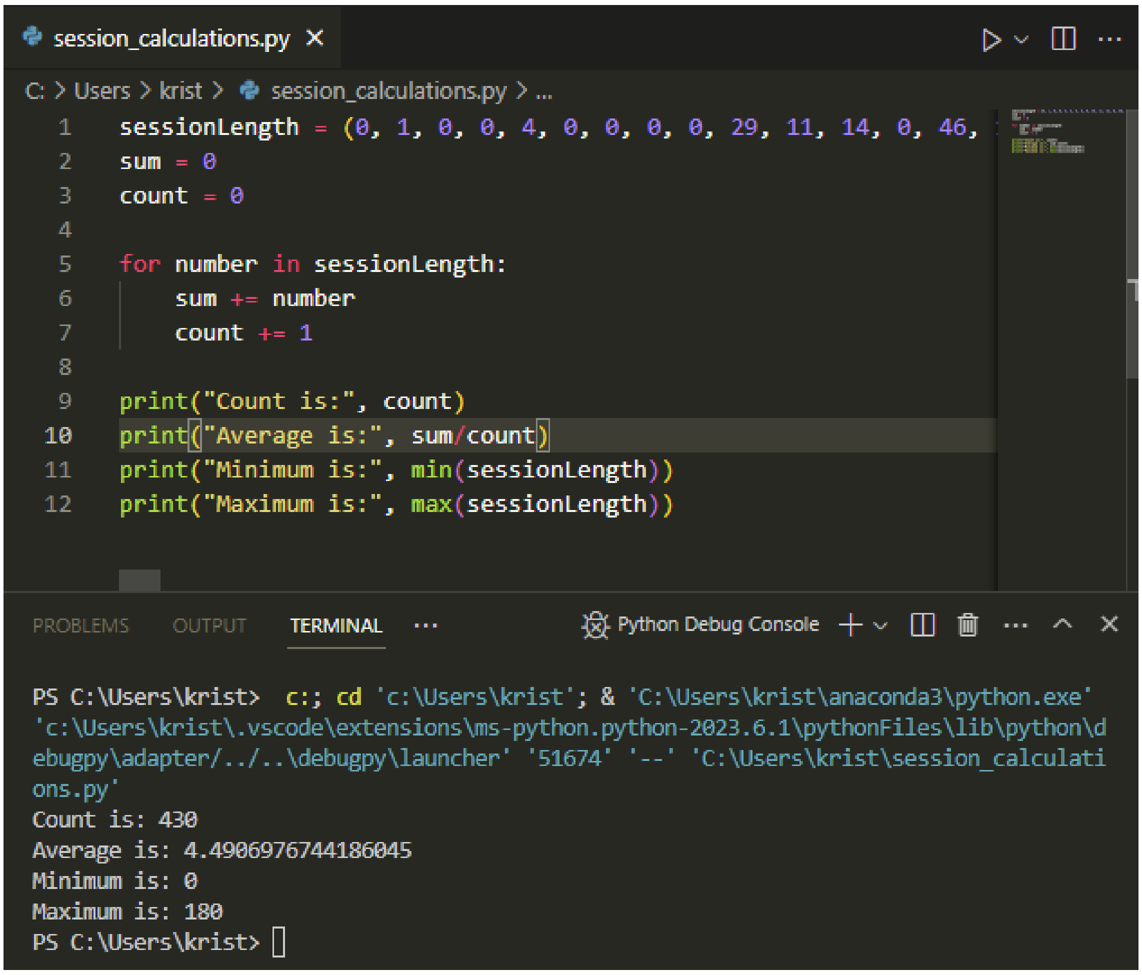


Figure 14. Graph of unique connections from Cowrie

This data was gathered by applying a filter to all the logs, which contained the keyword “new connection.” The data captured by the honeypot can help improve the accuracy of SOCs detection capabilities by reducing false positives and identifying previously unknown threats. With a more accurate detection system, the SOC can allocate resources more effectively and respond to genuine threats more efficiently.

The connection data was downloaded from Opensearch, and session length was extracted from the data. A small python script was written to calculate the average, minimum and maximum session length, see Figure 15:



```python
sessionLength = (0, 1, 0, 0, 4, 0, 0, 0, 0, 29, 11, 14, 0, 46,
sum = 0
count = 0

for number in sessionLength:
    sum += number
    count += 1

print("Count is:", count)
print("Average is:", sum/count)
print("Minimum is:", min(sessionLength))
print("Maximum is:", max(sessionLength))
```

```
PS C:\Users\krist>  c:; cd 'c:\Users\krist'; & 'C:\Users\krist\anaconda3\python.exe'
'c:\Users\krist\.vscode\extensions\ms-python.python-2023.6.1\pythonFiles\lib\python\d
ebugpy\adapter/../..\debugpy\launcher' '51674' '--' 'C:\Users\krist\session_calculati
ons.py'
Count is: 430
Average is: 4.4906976744186045
Minimum is: 0
Maximum is: 180
PS C:\Users\krist>
```

Figure 15. Python script for calculating minimum, maximum and average session length.

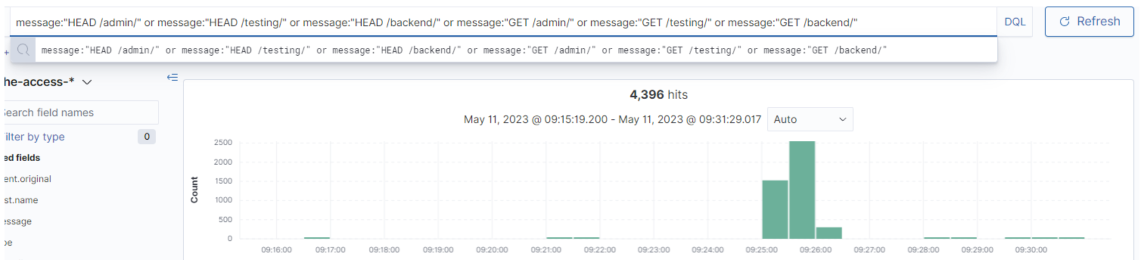
This analysis reveals that the average session length between the Cowrie honeypot and the adversary system is 4.49 seconds. This short duration can be attributed to the fact that many of the cyberattacks performed on the system were conducted by automated tools or scripts, which can initiate and complete their tasks rapidly. Automated attacks are common in the cybersecurity landscape, as they allow adversaries to target countless systems with minimal effort and time investment.

However, the maximum session length of 180 seconds suggests that a human attacker was interacting with the honeypot during some instances. Human adversaries often spend more time exploring, probing, and attempting to compromise the system, as they rely on manual techniques and decision-making processes to adapt their approach based on the system's responses. In these cases, the longer session length can provide valuable insights into the attacker's behavior, skill level, and objectives.

The difference in session lengths between automated and human-driven attacks underscores the importance of understanding and adapting to the diverse types of threats faced by organizations. By analyzing the session lengths and other behavioral patterns in the Cowrie honeypot, security teams can gain a better understanding of the nature of the attacks, identify trends, and develop appropriate countermeasures to protect against both automated and human-driven cyber threats.

Also, the effectiveness of using DirBuster to generate events for accessing fake directories and filtering logs in OpenSearch using the Lucene query language was evaluated. The objective was to assess the potential of this approach for improving SOC metrics and providing valuable intelligence on potential threats.

Fake directories were created using examples obtained from the official Kali Linux Packages GitHub repository, and two types of DirBuster scans were conducted - a recursive scan and a normal scan. These scans were used to simulate attacks and generate events for accessing the fake directories. To capture these events, a query in Lucene query language was created to filter the logs and only capture access events related to the fake directories, see Figure 16:



```
message:"HEAD /admin/" or message:"HEAD /testing/" or message:"HEAD /backend/" or message:"GET /admin/" or message:"GET /testing/" or message:"GET /backend/"
```

4,396 hits

May 11, 2023 @ 09:15:19.200 - May 11, 2023 @ 09:31:29.017 Auto

Figure 16. Lucene query for filtering fake directory access logs

This approach enabled us to easily distinguish between legitimate and malicious traffic. The results suggest that events, which are generated when accessing fake directories, and filtering logs in OpenSearch using the Lucene query language can be an effective approach for improving SOC metrics and providing an early insight into potential threats.

Overall, the data obtained from the Cowrie honeypot, along with the use of fake directories within the deception environment can contribute to improving several SOC metrics.

Firstly, it can enhance detection capabilities by identifying new attack vectors and methods used by adversaries. This information enables the refinement and expansion of detection rules and signatures, thereby increasing the accuracy of threat detection mechanisms.

Secondly, the data analysis allows for a better understanding of attacker behaviours and motivations, aiding in the identification of potential threat actors and their tactics. This knowledge can facilitate more precise threat intelligence and enable targeted threat hunting activities, resulting in improved incident response and mitigation.

Lastly, the analysis of the data from fake directories enables the identification of potential reconnaissance activities. Analysing the requests to the fake directories allows for the detection of scanning or probing behaviours, indicating potential attackers attempting to gather information about the organization's infrastructure. This knowledge enhances the accuracy of SOC metrics related to reconnaissance detection and helps identify potential threats at an early stage.

6 Conclusions and recommendations

In conclusion, this research aimed to design and develop a prototype SOC system that leverages deception technology to improve the accuracy of security metrics, with a specific focus on the reconnaissance phase of the cyber kill chain model. The prototyped system was designed to detect and analyse reconnaissance attempts using deception techniques such fake directories, and fake services.

The effectiveness of the prototype SOC system was evaluated by collecting and analyzing data on its performance in detecting and responding to reconnaissance attempts. The data collected from the experimental setup showed that the use of deception technology can be used to improve the accuracy of SOC metrics related to the reconnaissance phase.

The author deems that the research questions raised in the first chapter of the thesis were answered.

1. What are the challenges in using real security incidents to improve SOC metrics?

Using real security incidents to improve SOC metrics can present several challenges. One of the primary challenges is the potential impact on production systems and critical assets. Another challenge is the variability and unpredictability of real incidents. Each incident may have unique characteristics, making it difficult to establish consistent metrics and benchmarks. Additionally, organizations may encounter limitations in obtaining comprehensive incident data due to the inability to detect and capture all incidents.

2. How can organizations improve the accuracy and completeness of data used to measure SOC metrics?

Organizations can leverage deception technology as a potential solution to improve the accuracy and completeness of data used to measure SOC metrics. Deception technology offers several benefits in this regard:

a) Enhanced Data Collection: By deploying deceptive elements such as fake directories, or decoy systems, organizations can attract and capture the attention of potential attackers. This enables the collection of real-time, targeted data about attack patterns, techniques, and IOCs. This rich dataset can significantly enhance the accuracy and completeness of the data used for measuring SOC metrics.

b) Controlled Environment: Deception technology allows organizations to create a controlled environment solely for the purpose of gathering security-related data. By segregating deceptive elements from production systems, organizations can ensure that the captured data is isolated and uncontaminated, leading to more accurate measurements of SOC performance.

c) Early Threat Detection: Deception technology acts as an early warning system by attracting and engaging potential attackers. As attackers interact with the deceptive elements, organizations can detect and respond to threats at an earlier stage, mitigating the impact and reducing the time to remediation. This proactive approach improves the accuracy and timeliness of the data used for measuring SOC metrics.

3. What are the key considerations when implementing deception technology?

When implementing deception technology, organizations should consider the following key aspects:

a. Clear Objectives: Define clear goals and objectives for deploying deception technology. Identify specific use cases, such as diverting attackers, gathering intelligence, or improving SOC metrics, to guide the implementation process effectively.

b. Realism and Relevance: Create deceptive elements, such as fake directories, that resemble the real environment to make them convincing to potential attackers.

c. Integration and Monitoring: Integrate deception technology with existing security systems, such as SIEM, intrusion detection systems, or threat intelligence platforms, to centralize monitoring and analysis. The findings of this thesis work highlight the significant potential of integrating deception technology with SIEM systems.

d. Continuous Improvement: Regularly assess and update the deception strategy to adapt to evolving attack techniques and attacker behaviors. The findings highlight the significance of a proactive and adaptive approach, allowing SOC teams to stay ahead of attackers and continuously enhance their cybersecurity defenses.

Based on the findings of this research, it is recommended that SOC systems incorporate deception technology to improve the accuracy of security metrics, especially in the reconnaissance phase of the cyber kill chain model. Deception technology can be used to create decoys that mimic real systems, services, and data, which can mislead attackers into revealing their presence and intentions. This approach can help security teams to detect and respond to reconnaissance attempts more accurately and efficiently.

In summary, the analysis of data from the Cowrie honeypot, alongside the use of fake directories, enhances SOC metrics by improving threat detection, incident response, and overall cybersecurity effectiveness. This integrated approach provides a comprehensive understanding of attacker behaviors and allows for targeted improvements in security operations.

It is recommended that future research focuses on evaluating the effectiveness of deception technology in other phases of the cyber kill chain model, such as weaponization, delivery, exploitation, installation, command and control, and actions on objectives. Additionally, more research is needed to evaluate the effectiveness of different types of deception technology, such as honeypots, honeytokens, and honeyfiles, and how they can be integrated into SOC systems to improve security metrics.

7 References

[1] J. Sadowski and C. Charrier, “Mandiant,” 20 03 2023. [Online]. Available: https://www.mandiant.com/resources/blog/zero-days-exploited-2022. [Accessed 01 04 2023].

[2] Ponemon Institute, “Improving the Effectiveness of the Security Operations Center,” 06 2019. [Online]. Available: https://www.devo.com/wp-content/uploads/2019/07/2019-Devo-Ponemon-Study-Final.pdf. [Accessed 09 04 2023].

[3] R. Khudhair and A. Abbas, “ResearchGate,” 11 2016. [Online]. Available: https://www.researchgate.net/profile/Rana-Khudhair-Ahmed/publication/311325357_Security_Metrics_and_the_Risks_An_Overview/links/5841c26808ae61f75dd0f9dd/Security-Metrics-and-the-Risks-An-Overview.pdf. [Accessed 09 04 2023].

[4] E. Agyepong, Y. Cherdantseva, P. Reinecke and P. Burnap, “Challenges and performance metrics for security operations center analysts: a systematic review,” 09 12 2019. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/23742917.2019.1698178. [Accessed 23 04 2023].

[5] FortiNet, “FortiNet, Deception Technology,” FortiNet, [Online]. Available: https://www.fortinet.com/resources/cyberglossary/what-is-deception-technology. [Accessed 23 04 2023].

[6] R. M. Amin, M. J. Cloppert and E. M. Hutchins, “Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains,” January 2011. [Online]. Available: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf. [Accessed 23 04 2023].

[7] M. H. Almeshekah and E. H. Spafford, “Cyber Security Deception,” 16 07 2016. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-32699-3_2. [Accessed 23 04 2023].

[8] K. Ferguson-Walter, T. Shade, A. Rogers, M. Trumbo, K. Nauer, K. Divis, A. Jones, A. Combs and R. Abbott, “The Tularosa Study: An Experimental Design and Implementation to Quantify the Effectiveness of Cyber Deception,” 01 05 2018. [Online]. Available: https://www.osti.gov/servlets/purl/1524844. [Accessed 23 04 2023].

[9] K. J. Ferguson-Walter, M. M. Major, C. K. Johnson and D. H. Muhleman, “Examining the Efficacy of Decoy-based and Psychological Cyber Deception,” 11–13 August 2021. [Online]. Available: https://www.usenix.org/system/files/sec21-ferguson-walter.pdf. [Accessed 23 04 2023].

[10] Wikipedia, “OSSIM,” 26 09 2022. [Online]. Available: https://en.wikipedia.org/wiki/OSSIM. [Accessed 24 04 2023].

[11] “OpenSearch,” GitHub, 19 04 2023. [Online]. Available: https://github.com/opensearch-project/OpenSearch. [Accessed 24 04 2023].

[12] Cowrie, “Github,” [Online]. Available: https://github.com/cowrie/cowrie. [Accessed 24 04 2023].

[13] “GitLab,” 09 04 2023. [Online]. Available: https://gitlab.com/kalilinux/packages/dirbuster/. [Accessed 11 05 2023].

[14] J. Reed, “SecurityIntelligence,” SecurityIntelligence, 31 August 2022. [Online]. Available: https://securityintelligence.com/news/40-percent-zero-day-exploits-decade-2021/. [Accessed 01 12 2022].

[15] P. Dholakiya, “What Is the Cyber Kill Chain and How It Can Protect Against Attacks,” [Online]. Available: https://www.computer.org/publications/tech-news/trends/what-is-the-cyber-kill-chain-and-how-it-can-protect-against-attacks. [Accessed 23 04 2023].

Appendix 1 - Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Kristo Tammsoo

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Improving SOC Metrics Using Deception Technology” supervised by Mohammad Tariq Meeran and co-supervised by Madis Männik.

   1.1 to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;

   1.2 to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.

2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.

3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

14.05.2023

[1] The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – OpenSearch cluster docker-compose.yml

```yaml
version: '3'

services:
  opensearch-node1:
    image: opensearchproject/opensearch:latest
    container_name: opensearch-node1
    environment:
      - cluster.name=opensearch-cluster
      - node.name=opensearch-node1
      - discovery.type=single-node
      - bootstrap.memory_lock=true
      - "OPENSEARCH_JAVA_OPTS=-Xms1024m -Xmx1024m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
      nofile:
        soft: 65536
        hard: 65536
    volumes:
      - opensearch-data1:/usr/share/opensearch/data
    ports:
      - 9200:9200
      - 9600:9600
    networks:
      - opensearch-net
  opensearch-dashboards:
    image: opensearchproject/opensearch-dashboards:latest
    container_name: opensearch-dashboards
    ports:
      - 5601:5601
    expose:
      - "5601"
    environment:
      OPENSEARCH_HOSTS: '["https://opensearch-node1:9200"]'
    networks:
      - opensearch-net

volumes:
  opensearch-data1:

networks:
  opensearch-net:
```

Appendix 3 – LogStash Apache access log configuration

```
input {
  file {
    path => "/var/log/apache2/access.log"
    type => "apache-access"
  }
}

output {
  opensearch {
    hosts => ["https://192.168.10.20:9200"]
    index => "apache-access-%{+YYYY.MM.dd}"
    user => "apache"
    password => "************"
    index => "syslog-%{+YYYY.MM.dd}"
    ssl => false
    ssl_certificate_verification => false
  }
}
```

Appendix 4 – LogStash Cowrie log configuration

```
input {
  file {
    path => "/home/kristo/cowrie/var/log/cowrie/cowrie.log"
    type => "syslog"
  }
}

output {
  opensearch {
    hosts => ["https://192.168.10.20:9200"]
    user => "cowrie"
    password => "***********"
    index => "syslog-%{+YYYY.MM.dd}"
    ssl => false
    ssl_certificate_verification => false
  }
}
```