5. appendix **COVER SHEET**

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

JOHN VON NEUMANN FACULTY OF INFORMATICS

THESIS

OE-NIK 2022-23

Name of the student: **Aleksa Jankovic**
Identification number of the student: **T/006069/FI12904/N**

Óbudai University
John von Neumann Faculty of Informatics
Institute for Cyber-physical Systems

THESIS WORK SPECIFICATION

Name of the student:	**Aleksa Jankovic**
Identification number of the student:	T/006069/FI12904/N
Neptun's code:	[redacted]

Title of the thesis:

Detecting and preventing cyberattacks using Honeytokens
Kibertámadások megelőzése és kivédése Honeytoken-ek segítségével

Internal supervisor	Dr. Eszter Balázsné Kail
External supervisor:	Mihály Zágon
Deadline:	15 December 2022
Subjects of the final exam:	Computer Architectures Cloud service technologies and IT security specialization

The task:

While honeypots can be fake servers or other types of resources, honeytokens are fake data that attackers unknowingly can take with them, thus revealing information about their activities. In this way honeytokens can be used to detect and to track malicious activities. They seem to be very attractive to attackers, but actually they are useless to them. This can help IT teams to prevent further escalation into the network and to prevent future incidents.
The goal of the thesis is to identify and generate appropriate honeytokens that could be useful in the detection of the attacks.
The task of the student is to design, generate and place honeytokens into the system in order to detect and alert attackers' action. The student should also test its operation by generating attacks and handling alerts.

The thesis must contain:

- a thorough literature review,
- description of the honey token based detection technique,
- review about attack tactics,
- design of the test environment and test cases,
- analysis of the results,
- conclusion and future development ideas.

Place of stamp

Fleiner R.
.....
Dr. Rita Fleiner
head of institute

This specification is valid until: **15 December 2024**
(According to OE TVSz 55.§)

I consider the thesis suitable for submission:

.....
external supervisor

Kail
.....
internal supervisor

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics
Annex 7

STUDENT'S DECLARATION

I the undersigned student hereby declare that this degree project / thesis is the result of my own work; I have disclosed the references and tools used in an identifiable manner. The results indicated in my degree project / thesis completed may be used by the university and the institution announcing the task for their own purposes free of charge.

Dated: Budapest, 15 December 2022

.....
student's signature

CONSULTATION LOG

Student's name:	Neptun code:	Specialty:
Aleksa Jankovic	[redacted]	**Computer engineering (BSc)**

Phone number: Mailing address (e.g. domicile):

[redacted]

Degree project / thesis[1] title in Hungarian:

Kibertámadások megelőzése és kivédése Honeytoken-ek segítségével

Degree project / thesis[2] title in English:

Detecting and preventing cyberattacks using Honeytokens

Institutional supervisor:	External supervisor:
Balázsné Dr. Kail Eszter	Mihaly Zagon

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	06.10.2021	Discussion over thesis topic	Kail
2.	20.10.2021	Paperwork and structure	Kail
3.	03.11.2021	Views at first draft and talk about the direction of the implementation	Kail
4.	17.11.2021	Presentation	Kail

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, 24.11.2021

Kail

.....

institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics
Annex 9

CONSULTATION LOG

Student's name: **Aleksa Jankovic**

Neptun code: [redacted]

Specialty: **Computer engineering (BSc)**

Phone number: [redacted]

Mailing address (e.g. domicile): [redacted]

Degree project / thesis[1] title in Hungarian:

Kibertámadások megelőzése és kivédése Honeytoken-ek segítségével

Degree project / thesis[2] title in English:

Detecting and preventing cyberattacks using Honeytokens

Institutional supervisor: **Balázsné Dr. Kail Eszter**

External supervisor: Mihaly Zagon

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	14.09.2022	Discussion over thesis topic	Kail
2.	19.10.2022	Problems	Kail
3.	07.11.2022	Final version	Kail
4.	05.12.2022	Presentation	Kail

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, 24.11.2022

Kail

.....
institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

Contents

1. Introduction

Honeytokens are fake resources or data that can be used to detect and track malicious activities. They seem to be very attractive to attackers, but actually, they are useless to them. At the same time, honeypots may be a fake servers or other types of resources that tell us only if the attack has occurred. Honey tokens are fake data that attackers unknowingly can take with them, thus revealing more detailed information about their activities. This can help IT teams, to prevent further escalation into the network and prevent future incidents.

The goal of the thesis is to identify and generate appropriate honeytokens that could be useful in the detection of attacks. The task is to design, generate and place honeytokens into the system in order to detect and alert attacker's actions. As well as provide the result of using the monitoring tool Sysdig Falco.

In this paper, the topic "Detection and prevention of attacks using Honeytoken" will be discussed. The initial chapters will discuss what honeytokens actually are and what types of honeytokens there are. After that, we will talk about the HoneyGen method, which is used for the automatic generation of honeytokens, and what all the stages are in the HoneyGen method. After the description of the HoneyGen method, two-factor authentication will be discussed.

The chapter dealing with two-factor authentication describes what authentication factors are, how two-factor authentication works, what types of two-factor authentication exist, two-factor authentication for mobile phones, whether two-factor authentication is secure, and the mechanism of two-factor authentication with the help of a honeytoken.

Then the different types of MITER tables will be described. Furthermore, cyber-attacks and system exploitation will be described in the paper. Following up with a look at Sysdig Flaco and its deployment. And as the main topic of the work, in the practical part, we will simulate attacks and how the honeytoken will react to such attacks as well as which information the honeytokens will provide for the defender.

2. What are Honey Tokens?

The term honey token refers to data that may be attractive to cybercriminals but is utterly useless for them. This term refers to a fake IT resource made and placed in a system or network so that cybercriminals can attack it. Henceforth why honeytokens are similar to honeypots.

The main difference between honeypots and honey tokens is that honey pots represent fake servers or other types of resources. In contrast, honey tokens contain data that attackers take with them, unwittingly revealing necessary information that helps IT teams prevent future attacks or detect the attacker.

Honey tokens contain data created and monitored to catch cybercriminals. This data contains markers that may appear to be a valuable asset to a cybercriminal but, in reality, have no real value to the organization. Once an attacker retrieves this data, the stolen data could contribute to tracking the attacker, whether they launched the attack inside or outside the organization.[1]

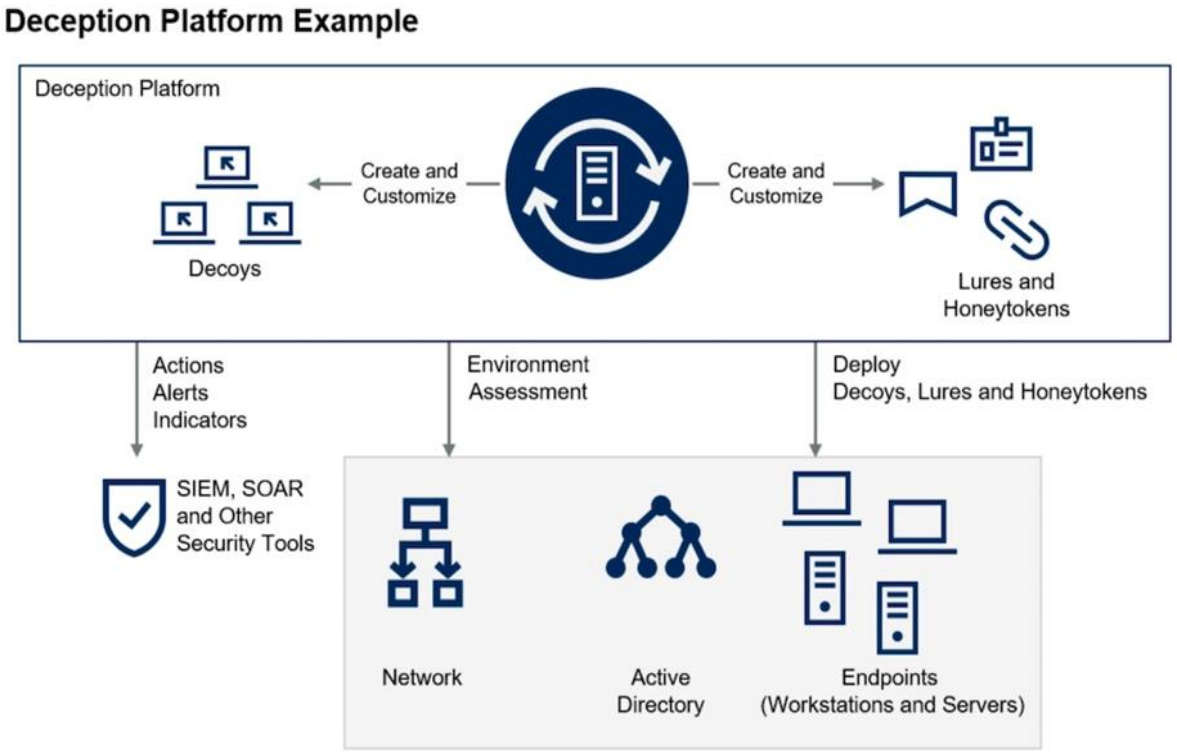

Figure 1: Honeytokens[2]

2.1. What Is a Honeypot vs Honey Token?

In the world of cyber security, organizations that take measures to protect against cyber criminals tend to take a defensive stance, which means engaging in a high-tech pursuit of finding and catching cyber criminals. Honey tokens and honeypots represent different approaches for detecting and catching cyber criminals. Organizations use them proactively by tricking cybercriminals into revealing information about themselves and the methods they use. A honeypot is a fake system placed next to a real digital asset. The purpose of a honeypot is to look very attractive to the cybercriminal. If the thief falls for the bait, he spends his valuable resources on a fake system or fake data and reveals the necessary information about his attack strategy.[1]

Honey tokens, unlike honeypots, are more of a way to identify cyber criminals. They are mainly used to track attackers, extracting information about their identity but also revealing the methods they use to exploit the system.

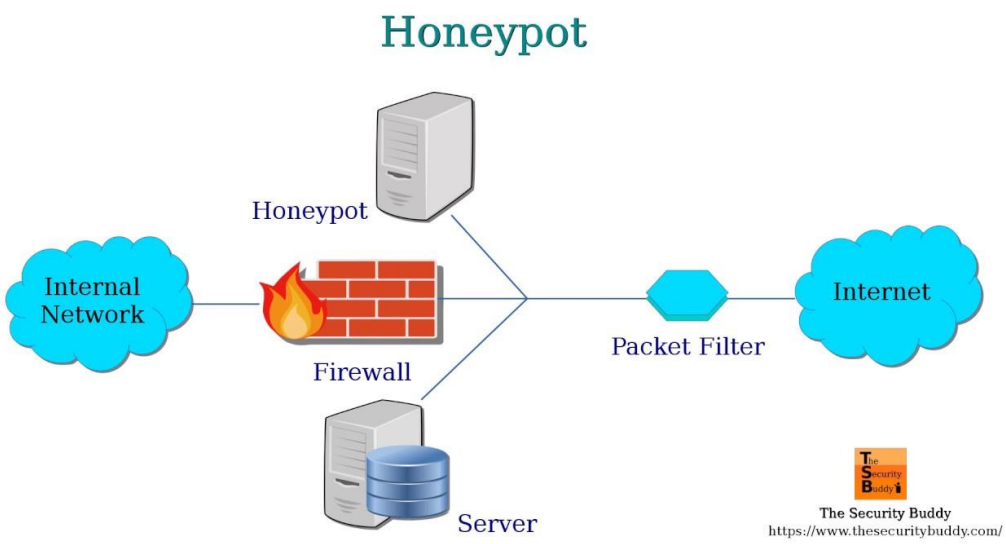

Figure 2: Honeypot[3]

2.2. Honeywords

The Honeywords scheme is based on customizing the password storage mechanism in such a way that, along with the password, a string of fake passwords is associated with each

account. Honeywords is a name for fake passwords. Sweet words are a combination of honey words and passwords. When entering a password during the authentication process, it can be immediately discovered that the database containing the passwords has been compromised. Unlike some traditional methods, implementations based on honeywords can effectively detect intrusions into databases containing passwords.[4]

The method behind Honeywords is as follows: During the authentication process, users can choose a username and password, similar to other forms. The login server (LS) then produces the password honeywords and maintains a record of the password databases. LS chooses the order of honeywords randomly in each subsequent reform, and also LS sends the user ID and actual index of each password to the Honeychecker (HC). Honeychecker is a utility server built to store password indexes.

Initially, Honeywords was built with the expectation that an attacker could steal the hashed passwords and invert the hashes to obtain the passwords. It was also assumed that the attacker would not abuse LC and HC in the same time frame. Honeywords serve to protect passwords from brute force and dictionary attacks. This method attempts to prevent intrusion into a database containing passwords. It only attempts to prevent offline attacks against the dictionary, where the assumption is that the attacker only stole the password hashes and then left the system.[4]

3. Types of Honey Tokens

There are different variations of honey tokens, and each interpretation is designed for a specific application.

3.1. Fake E-mail Addresses

Fake e-mail addresses are attractive to thieves, tricking them into revealing important information about themselves, such as personal information, usernames, and passwords, as well as important information about the organization they work for. We can set up a fake e-mail account using randomly generated names or even the names of some famous people that the attackers do not know. Once fake e-mail accounts are set up, they are left inactive, but it must be ensured that attackers can detect them on the e-mail server or some other location that attackers can easily access.[1]

Since fake e-mail accounts are not used, the only way to attack is through phishing e-mails or spamming by someone who has managed to hack an organization's internal server or web server and thus gain access to e-mail addresses. Fake passwords can also be used to lure attackers.[1]

3.2. Fake Database Data

Fake database data are fictitious records that can be attractive to cybercriminals. If the attacker falls for the bait, he steals the data but also takes the honey token with him. This is an excellent way to pinpoint a particular weakness that allows a cybercriminal to enter a specific organization's system. You can also find other types of loopholes, such as an error – or a calculated move – at the management level[1]

3.3. Fake Executable Files

Fake executables are similar to malware but are created by organizations and used against attackers. These are usually applications or other software programs containing a switch that sends a signal back after the attacker triggers them. Once activated, the switch can provide essential information, such as the hacker's IP address and the usernames associated with the hacker's system.

Fake executable files can damage the attacker's system if he does not protect his machine from this attack. For example, if an attacker blocks their external ports when running a program, the information the organization needs will not be returned.[1]

3.4. Web beacons

A web beacon is a link to an object within a file. This file can be tiny and difficult to see by the naked eye, such as a graphic with only one pixel in size. After opening the document containing the beacon, the web beacon sends data to the IT team with essential details about the attacker's system and its location on the Internet.

Web Beacon has a similar problem to fake executable files. If the attacker has a firewall on his machine, which prevents outgoing information from being transmitted, the organization using this defense system will not receive feedback.[1]

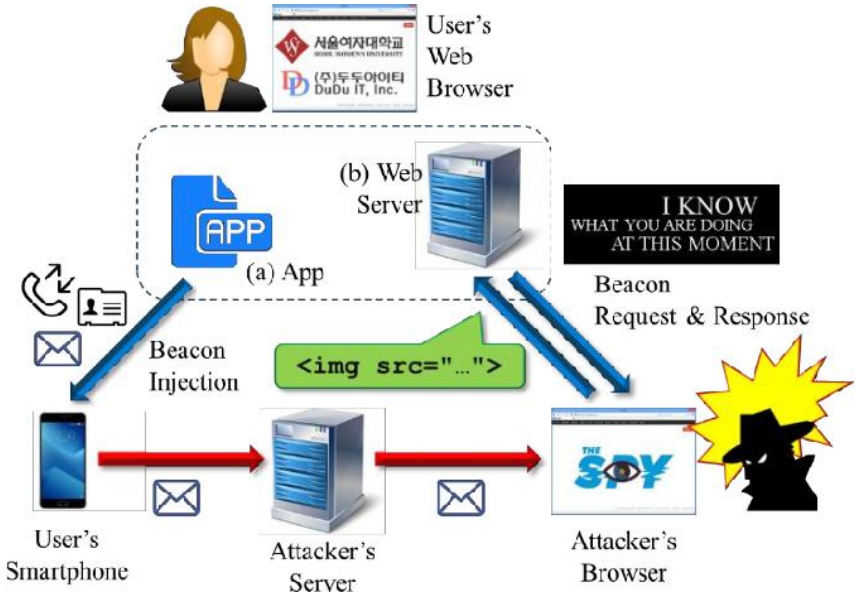

Figure 3: Web beacons[5]

3.5. Browser Cookies

With web browser cookies, it is possible to bypass hackers' attempts to neutralize strategies such as rogue executables and web, which depend on external ports being open for transmission. Web browser cookies provide information about hackers' online activities, similar to how Google or Bing collect information about their visitors.

This method is successful in most cases, as the hacker or someone else in his network can often forget to clear his web browser cache or otherwise hide his online activities. This is a relatively common mistake and can often leave hackers open to sharing sensitive information about their activities.[1]

3.6. Canary Traps

The canary trap has been established from the idea that they can act as "canary", a spy basically, by "singing" meaningful information about the organization. The honey token is placed in a file that will be attractive to someone who wants to steal the information. For example, you can put a honey token in a false accounting spreadsheet that reveals confidential information regarding the company's finances.[1]

A honey token can also be placed in a legitimately valuable file, knowing that there is someone in the organization to whom the information may be leaked. For example, the organization can put the honey token inside an internal bulletin, which is only received by the company employees. Every one of these bulletins can contain an identifier installed in it, which is special to the person who receives it. After the canary sends essential information to someone outside the organization, the token can help identify the mole inside the organization.[1]

3.7. AWS Keys

Amazon Web Services (AWS) uses digitally signed keys to unlock certain areas within an organization's infrastructure designed to restrict access. Keys can be placed in various locations, such as the desktop, in text files, within GitHub repositories, and various other places.

Cybercriminals are naturally attracted to AWS keys for a variety of reasons. For example, an AWS key can be used to gain complete control over a company's entire infrastructure. If a hacker obtains one of these keys, he can gain administrative privileges and make various differences in the applications and data flow. This method can be used to sabotage the system as well as divert valuable digital assets to the hacker. Suppose an attacker is unable to obtain a key that grants administrative privileges. In that case, a key with lower-level credentials can still be used by the attacker to steal other important information, increase his privileges, or break into other aspects of the system.[1]

An organization can use the hacker's wishes against them, and using AWS keys as honey tokens is very simple. If we want to use the AWS key, it must be tested first. AWS keys issued by Amazon have a logging mechanism built into them. When a hacker tests this key, their activities are logged, so it is up to the IT team to deploy fake keys.[1]

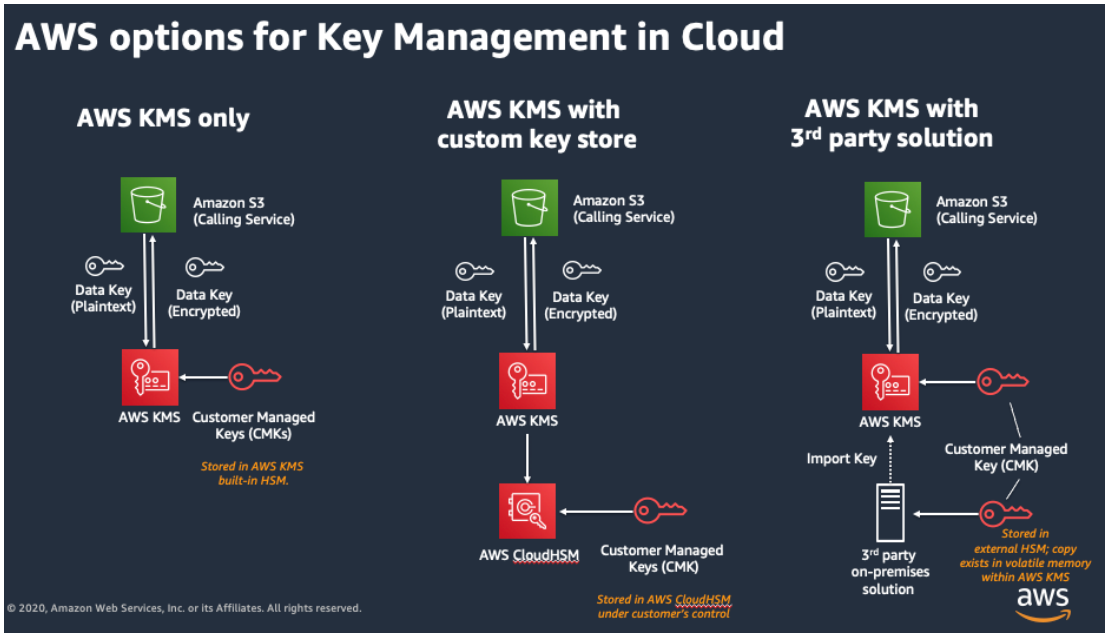

Figure 4: AWS keys[6]

4. HoneyGen Automated Honeytokens

This topic presents the new automated method HoneyGen, which serves to generate generic honeytokens. HoneyGen has adopted the characteristics and properties of real data, to be able to create honeytokens, which have a great similarity with real tokens, and even in many cases are indistinguishable from real tokens.

Honeytokens are artificial digital data items, which are deliberately inserted among real system resources, in order to detect unauthorized attempts to use essential information. Due to their characteristic properties, honeytokens appear as original data items. Honeytokens are also characterized by their availability to potential attackers, who intend to breach the security of an organization, with the aim of extracting some important information. One of the biggest challenges when generating honeytokens is creating data items that look like real data and are hard to distinguish from real tokens.

In this chapter, the HoneyGen method, which is used for automatic honeytoken generation, will be presented. HoneyGen, by extrapolating the characteristics and properties of real data items, creates honeytokens that are very similar to real data.[7]

Three main stages when generating a honeytoken:

- Rule mining; At this stage, various rules characterizing real data are extracted from production databases.
- Honeytoken generation; At this stage, an artificial relational database is generated based on the extracted rules.
- Assessment of probability; At this stage, a score is calculated for each honeytoken individually based on its similarity to real data.

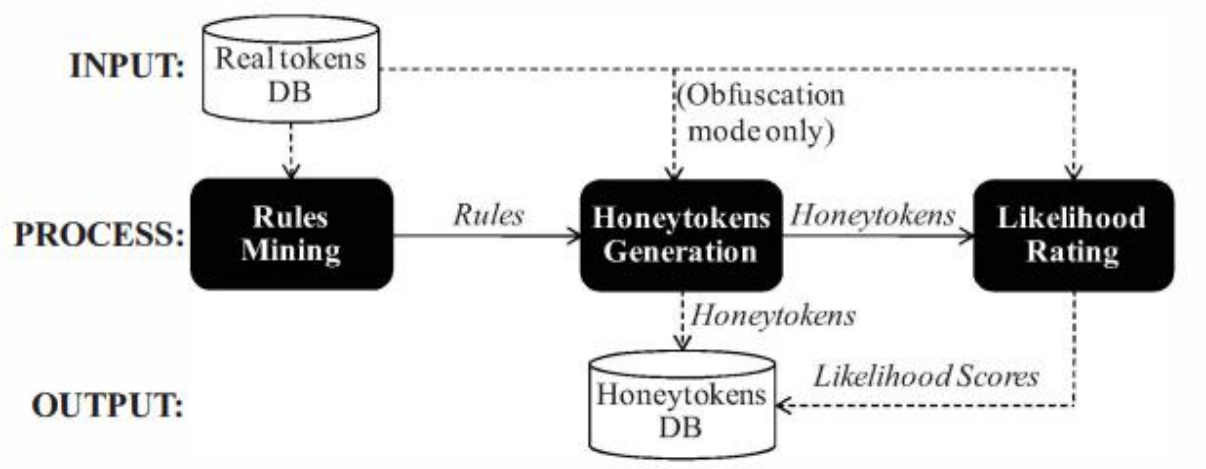

Figure 5: HoneyGen's three phases: 1. Rule Mining, 2. Honey token Generation, and 3. Likelihood Rating[7]

Because this is a fully automated honeytoken generation process, no human interventi on is required after defining the input. In the future, it is planned to develop a user interface, which will give users the possibility to influence the generation process. The user interface will provide the user who is responsible for the whole process, the ability to see the rules that have been automatically extracted, and thus the user is provided with the ability to edit, add and remove rules.

This will provide system administrators with many benefits, which is reflected in increased flexibility in the honeytoken generation process. Also, the user will be able to add new values or delete existing values found in the dimension tables. Future research is necessary to determine the required number of honeytokens for different scenarios, when they need to be added to the database, and when they need to be exchanged for new ones.[7]

5. Critical infrastructure networks

Critical network infrastructure is a term that is most often associated with national assets that are of great importance for the operational stability of the economy and society and without which, in the 21st century, a nation-state cannot be successfully run. Critical data infrastructure operations nowadays are powered by intelligent and sophisticated networks called critical infrastructure networks. In today's time, there is a vast number of critical infrastructures.[8]

Critical infrastructures that appear most often are:

- Electrical network
- Oil and gas sector
- Nuclear power plants
- Water supply systems
- Air traffic systems
- Water treatment plants
- Railway traffic systems
- Industrial production

Critical infrastructure networks usually have command and control systems in place to run operations smoothly and efficiently. Supervisory control and data acquisition (SCADA) is most often used for command and control. SCADA collects data from all available systems with the help of a wide range of sensors. After the data collection is complete, commands are issued from the Master Terminal Station (MTU) to guide the proper operators. **Figure 6** shows a typical topology of a critical infrastructure sensor network.[8]

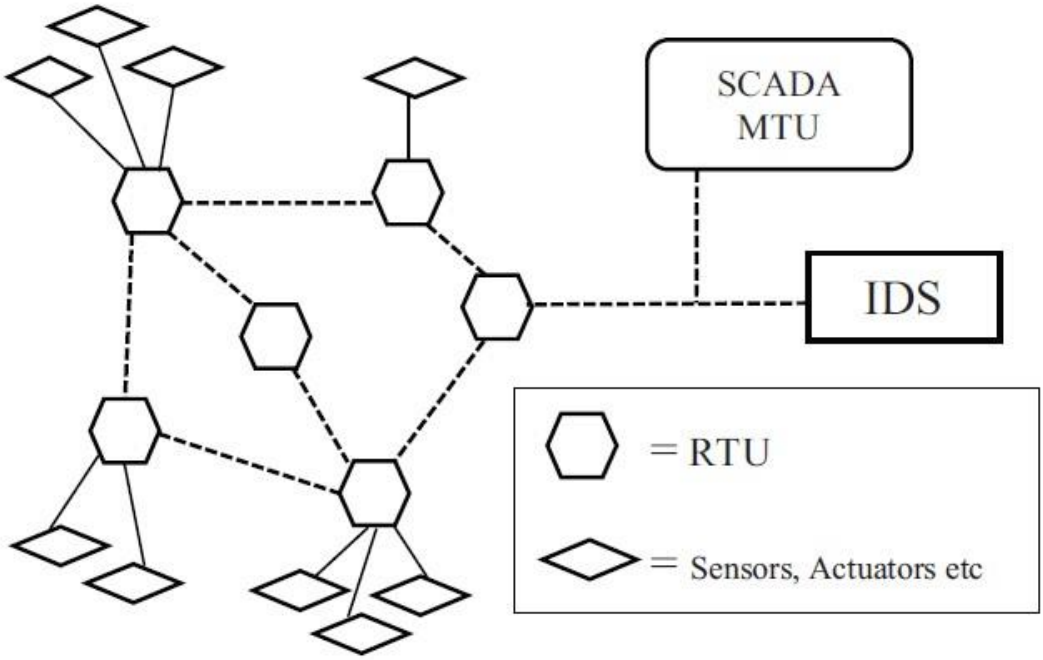

Figure 6: Critical Infrastructure Network Topology[8]

The SCADA system is connected to a network of routing nodes known as the Remote Terminal Unit (RTU). Furthermore, sensors are also connected to RTUs. **Figure 6** shows a network-based intrusion detection system (NIDS), which in this way, serves a critical infrastructure sensor network with its security services.[8]

6. Two-factor authentication

Two-factor authentication (2FA) is a security mechanism where users use two separate keys to verify their identity, and this type of authentication is often called two-step authentication or two-factor authentication. This type of authentication is undertaken to secure both the system user's credentials and the tools that are available to the user and can be used. Two-factor authentication provides a higher level of authentication than single-factor authentication (SFA), which requires the user to enter only one factor (usually a password). The strategy behind two-factor authentication is based on a mechanism that provides a password and another protection mechanism that may be a biometric factor or a security token, such as a face scan or a fingerprint, which provides double protection. Two-factor authentication introduced an additional level of protection in the authentication process because it is difficult for attackers to access computers or online accounts. After all, more is needed to know the password of the account that is being hacked in order to pass the authentication process. In order to monitor access to confidential applications and files, two-factor authentication is used, and online service providers are gradually introducing 2FA in order to protect their customers from hacker attacks, which have broken into the database containing passwords.[4]

6.1. What are authentication factors?

There are many ways in which the authentication process can be carried out. Many authentication mechanisms mainly rely on information-containing factors like traditional passwords, while two-factor authentication methods rely on either a possession factor or an inherence factor.

Authentication factors in approximate order of adoption:

1. The knowledge factor is when the user knows some information, such as a password or personal identification number (PIN).
2. The possession factor is when the user must accept some authentication request. An authentication request may involve the user attaching, e.g., an ID card, security key, or mobile phone number or entering an authentication code that will be displayed in a mobile phone application.
3. The factor of inherence includes the confirmation of identity, which is based on the entry of authentication confirmation which is based on the authentication confirmation based on the physical property of individuals, which includes the entry of some biometric element, such as a fingerprint, face scan, or speech recognition. A fingerprint can be entered using the fingerprint scanner; face scanning can be done with the help of a dedicated camera, while speech

recognition is done with the help of a microphone. These authentication methods are very commonly used. In addition to these methods, other biometric behavioral characteristics, such as the dynamics of pressing specific keys or gait variations, are also in use.

4. The location factor is most often based on the location from which the authentication process is attempted and can be implemented by limiting authentication to a specific location or, more commonly, by tracking the Internet Protocol (IP)-based geographic source of the authentication attempt.
5. The time factor of authentication allows the user to log in to the system in a precisely specified period of time, and outside of that period, access to the system is disabled.

It should be known that a large number of two-factor authentication systems rely on the first three authentication factors. In contrast, multiple authentications (MFA), based on two or more separate passwords for the most reliable and secure authentication, are used in systems where greater security is required.[4]

6.2. How does two-factor authentication work?

This chapter will describe the common process of a two-factor authentication system:

- A program or website asks the user to log in.
- The user then enters information they know - usually a username and password. After that, the server performs the matching, and the user is remembered in the system.
- The accessed website creates a special key for user authentication purposes in processes where no password is required. The key is processed using the authentication function and then verified by the site server.
- After this, the site server requests the user to start the second login phase. Through this step, users must confirm their identity and must show their identification such as, e.g., identification key, ID card, smartphone, or other mobile devices; this step is called the ownership factor.
- In the fourth stage, the user must enter the created one-time code.
- In the fifth stage, the user is authenticated and gets access to a program or website after entering all the necessary variables.

In technical terms, two-factor authentication is necessary to gain access to a device at anytime. However, using two variables that belong to the same group does not represent 2FA. However, it always represents SFA because it requires a password and a shared secret, which

belong to the same class of authentication factors. User ID and password are not the most reliable as far as SFA services are concerned. A problem with password-based authentication is that generating excellent and strong passwords requires awareness and increased attention. It is necessary to protect passwords from many internal attacks, such as, e.g., carelessness when saving notes containing login credentials, passwords that may be on an old hard drive, or a vulnerability that exists in social engineering. Passwords can often be vulnerable to external threats and popout hackers using brute force.[4]

An attacker most often breaks password-based protection mechanisms and steals fundamental corporate data, which includes the user's data. Passwords remain the most commonly used type of SFA due to their low cost, straightforward execution, and familiarity. Several challenge-response questions and stand-alone approaches to biometric authentication can provide a more reliable and secure SFA process.[4]

6.3. Types of two-factor authentication products

There are different types of equipment and different utilities used to implement 2FA – starting from smartphone apps to tokens to radio frequency identification (RFID) cards.[4]

It is possible to separate two-factor authentication devices into two groups:

- Tokens are given to users to use when logging in and
- Infrastructure or software that can detect or authenticate data entry, users who use tokens correctly.

Physical devices may be in the form of smart cards or key fobs. they can serve as authentication keys or reside in mobile or web applications that generate PIN codes for authentication. Authentication codes are, in many cases, generated by the server, and these codes are commonly known as one-time passwords (OTPs). They are recognized as authentic by the system or application. An authentication code is a short sequence associated with a computer, user, or account and is used only once as part of the authentication process.

An essential feature of 2FA is allowing the authenticated user access to authorized services. Consequently, one of the most critical roles of 2FA is to associate the authentication method with the authentication data of the entity.[4]

6.4. How 2FA hardware tokens works

A number of authentication approaches are supported by available hardware tokens for 2FA. The YubiKey, a small universal serial bus (USB) system, supports OTPs, encryption, and public key authentication. Also, Universal 2nd Factor (U2F) is a protocol developed by the FIDO Alliance and is a common hardware token.

When logging in users using YubiKey to an OTP supported website, such as the following services; Gmail, GitHub, or WordPress, users insert their YubiKey into their computer's USB port, enter their account password, select the YubiKey field, and then click the YubiKey icon. The YubiKey generates an OTP and enters it in the correct field. OTP is a one-time password consisting of 44 characters. Of those 44 characters, 12 characters represent a special ID that defines the authentication key associated with the account, while the remaining 32 characters contain information that is encrypted using a key generated during the user's first registration and known only to the computer and Yubico servers. After that, the OTP is sent to Yubico for further verification, and then the Yubico authentication server sends back information about whether the token is valid. For the user, the information factor is the password, while the possession factor is the YubiKey.[4]

6.5. Two-factor authentication for mobile device authentication

Smartphones for implementing 2FA provide very large possibilities, thus encouraging organizations to choose what suits them best. The camera on a mobile phone can be used to recognize a face, while the microphone can be used to recognize speech, while certain applications can recognize a particular user's fingerprint using the built-in fingerprint reader. Smart cell phones have built-in GPS and can check the geographic location as an additional security measure; in addition to all that, SMS can be used for authentication in order to receive authentication codes by text message. A trusted mobile phone number can also be used to receive automated phone calls, which is another way to authenticate. In order for a user to participate in 2FA, they must provide at least one trusted mobile phone number. Apps that support 2FA are available for Android and iOS phones, as well as Windows 10, allowing the mobile phone to function as a physical interface.[4]

6.6. Is two-factor authentication secure?

With two-factor authentication, some limitations include costs for purchasing, issuing, and handling tokens or cards. From the user's perspective, having more than one two-factor authentication method provides the ability to store a pair of tokens or cards that are reasonably suspected of being lost or stolen. Two-factor authentication improves system security because access privileges are independent of passwords, but two-factor authentication systems can only be robust as their weakest part. For example, hardware tokens depend on the security of their manufacturer.[4]

6.7. The proposed two-Factor HoneyToken Authentication (2FHA) Mechanism

In this chapter, an alternative method for authentication, which serves to improve the security system, will be presented. This method consists of a combination of two-factor authentication and honeywords in order to make it impossible for an attacker to bypass the system's authentication mechanism. Even in cases where the attacker has access to the device through which the token is received, for example, if he has cloned the SIM card located on the mobile phone, the 2FHA method has made authentication very difficult or even impossible. The architecture of the 2FHA prototype is shown in **Figure 7**.

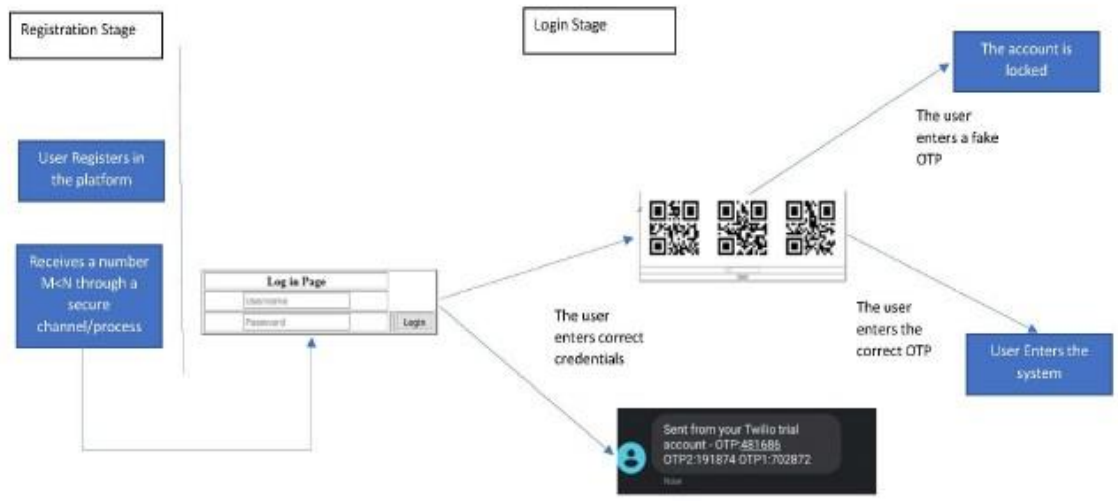

Figure 7: Architecture of the 2FHA prototype[4]

In order to demonstrate this method, a web page has been created, which includes a login page. If the user wants to log in to the system, he must first enter the correct username and password, which is the first authentication factor. The user's system then sends an M-number, which indicates a token, to be used at every future login. The user must enter the correct OTP if he wants to log in to the system from a new device. The user will receive a certain number of N-tokens and can choose the platform for which he will be alerted by receiving the token (by e-mail, SMS, phone call, etc.).

After that, the second authentication factor must be entered. The 2FHA mechanism generates three QR codes and a message that reaches mobile phone users. The SMS message contains all three OTP (one-time password) passwords, which correspond to each individual QR code. One password represents the correct password, while the other two are fake. The user must choose the most suitable method for filling the OTP field. It must be pointed out that in this example, the number of tokens produced is 3, but N tokens can be generated.

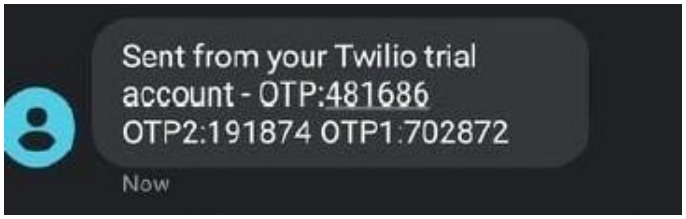

Figure 8: OTP passwords sent as an SMS message[4]

The process of scanning the QR code is very simple if the user chooses that method. Scan the correct QR code and then fill in the OTP field. QR code scanner is free software and mostly compatible with most devices. But if the user does not have a QR scanner, the convenient option is to use SMS. The SMS message shown in **Figure 8** is sent to the user at the time of his login to the system. As can be seen in **Figure 8**, there are three OTP passwords (OTP, OTP1, and OTP2) in the message. These passwords are created from QR codes, which are shown in **Figure 9**. Every user knows that only one of these three QR codes is valid, while the other two are fake.

Figure 9: The produced QR codes[4]

If the user has filled in the OTP field correctly, he will be able to enter the system, but if he makes a mistake during the login, he will be returned to the login page shown in **Figure 10** and will have to go through the login procedure from the beginning. As a preventive measure, the user's account may be suspended. When creating OTPs, certain rules must be followed; there must not be many similarities between them in order to avoid mistakes when writing.

Figure 10: The login page[4]

7. MITRE table analysis

7.1 MITRE table

MITRE table is a knowledge base which can be accessed globally, containing tactics and techniques of attackers which are collected from real-life experiences. Such a table of immensely valuable information serves its purpose to help private sectors, service community, cybersecurity products and even governmental set a foundation so they can develop defenses against specific threat methodologies and models. Naturely as a open source data base for many attacks and being used by big companies, it was a prime motivation for gather which attacks will be simulated in the implementation part. However before talking about the chosen attacks we first need learn what kind of attacks exist.[9]

7.2. Initial access

An organization's adversaries can exploit weaknesses in Internet-facing computers or programs by using specific software, data, or specific commands to cause unexpected or unintended system behavior. A weakness in a system can be an error that occurs, a malfunction, or a vulnerability in the design. Most often, these applications are websites, but they can also include databases (such as SQL), standard services (such as SMB or SSH), protocols for administration or management of network devices (such as SNMP or Smart Install), and all other applications that have access to the Internet, such as web servers or similar services. Depending on the flaw the attacker is trying to exploit, this may include Exploitation for Defense Evasion.

If the application is hosted on a cloud-based infrastructure and/or is containerized, exploiting that application may compromise the underlying instance or container. This can allow an attacker to access the cloud or container APIs, access the container host via Escape to Host, or exploit weak identity and access management policies.[9]

7.3. Execution

Adversaries can often abuse the container administration service in order to execute commands inside the container. Container administration services such as Docker daemon, Kubernetes API server, or kubelet can allow remote management of containers located in the environment.

In docker, you can set up an entry point during container set-up that can execute a specific script or command, or you can use the docker exec command inside a running container to execute a command. In Kubernetes, if an attacker has sufficient permissions, he can obtain

remote execution in a container in a specific cluster by interacting with the Kubernetes API server, kubelet, or by running the kubectl exec command.[9]

7.4. Persistence

Adversaries can exploit external remote services in order to access and/or persist in the network. Remote services such as VPN, Citrix, and many other access mechanisms can allow users from external locations to connect to internal network resources. There are often remote gateways that manage connections and authentication credentials for such services. Services that can also be used externally are VNC and Windows Remote Management.

Access to valid accounts for the use of a particular service is often a condition, which can most often be obtained through pharming credentials or by obtaining credentials from the user after the network has been compromised. There are two ways to use access to remote services, a redundant mechanism or a persistent access mechanism during operation.[9]

Access can also be obtained through an exposed service that does not require authentication. In containerized environments, this can include an exposed Docker API, a Kubernetes API server, a kubelet, or a web application such as a Kubernetes dashboard.[9]

7.5. Privilege Escalation

Adversaries can break out of the container to gain access to the underlying host. In this way, the adversary gains access from the host itself or to the host level to other container resources. In essence, container resources should provide a clear segregation of application functionality as well as be isolated from the host environment.

There are several ways in which the opponent can enter the home team's environment. Some examples are creating a container configured to mount the host's file system using bind parameters, which will allow an adversary to drop payloads while executing control utilities such as cron on the host or using a privileged container to run commands on the underlying host. Adversaries have the ability to escape via Exploitation for Privilege Escalation, exploiting a vulnerability in global symbolic links to gain access to the root directory of the host machine.

Acquiring a host gives our adversary opportunities to achieve some of the following goals, such as establishing persistence, lateral movement within the environment, or establishing a command or control channel on the host.[9]

7.6. Defense Evasion

Adversaries may use alternative authentication material such as password hashes, Kerberos tickets, and data access tokens to move laterally within our environment and bypass normal system access controls.

Authentication processes most often require a valid identity (e.g., username) in combination with one or more authentication factors (e.g., password, pin, physical smart card, token generator, etc.). Alternative authentication material can be legitimately generated by systems after a user or application is successfully authenticated by obtaining a valid identity and required authentication factors. Another way to generate alternative authentication material is during the identity creation process.

Caching the alternative authentication material allows the system to confirm that the identity has been successfully authenticated without requiring the user to re-enter the authentication factor or factors. The system must maintain an alternative authentication - in memory or on disk - there is a danger that the alternative authentication will be stolen through Credential Access techniques. If adversaries steal alternative authentication material, they will be able to bypass system access controls and authenticate even if they do not know plaintext passwords or other additional authentication factors.[9]

7.7. Credential access

Our adversaries can use brute force means to gain access to accounts if passwords are unknown or if password hashes are obtained. When passwords for an account or a set of accounts are anonymous, an adversary can systematically guess the password using an iterative or repetitive mechanism. Passwords can be brute-forced offline using previously acquired credential data, such as password hashes, or online by interacting with a server that verifies the legitimacy of those credentials.

It is possible for brute forcing of credentials to occur at various points during the breach. For example, adversaries of an organization may attempt to brute force access to valid accounts residing in the victim's environment using knowledge gathered from other post-compromise behaviors, such as discarding operating system credentials, account discovery, or password discovery. Adversaries combine brute force activities with behaviors such as External Remote Services as part of the Initial Access.[9]

7.8. Discovery

Our attackers may attempt to uncover our containers and other resources located in the container's environment. Other resources can represent images, implementations, modules, nodes, and additional important information, such as cluster status.

These resources within a web applications can be viewed: such as the Kubernetes dashboard or can be searched through Docker or Kubernetes APIs. In docker, logs can be a reason for the loss of information about the environment, such as the configuration of the environment, which services are available, and which cloud provider the victim can use.

Discovering these resources can inform us about the opponent's next steps in the environment, such as how the lateral movement is performed and what methods are used.[9]

7.9. Lateral Movement

Adversaries may use alternative authentication material such as password hashes, Kerberos tickets, and data access tokens to move laterally within our environment and bypass normal system access controls.

Authentication processes often require a valid identity (e.g., username) in combination with one or more authentication factors (e.g., password, pin, physical smart card, token generator, etc.). Alternative authentication material can be legitimately generated by systems after a user or application is successfully authenticated by obtaining a valid identity and required authentication factors. Another way to generate alternative authentication material is during the identity creation process.

Caching the alternative authentication material allows the system to confirm that the identity has been authenticated without requiring the user to re-enter the authentication factor or factors. The system must maintain an alternative authentication - in memory or on disk - there is a danger that the alternative authentication will be stolen through Credential Access techniques. If adversaries steal alternative authentication material, they can bypass system access controls and authenticate even if they do not know plaintext passwords or other additional authentication factors.[9]

7.10. Impact

Adversaries can execute Endpoint Denial of Service (DOS) attacks to corrupt or block the connection of services to users. Endpoint Denial of Service Endpoint attacks can be accomplished by exhausting the system resources on which services are hosted or exploiting the system to cause an endpoint state of failure. Some examples of services include websites, e-mail services, DNS, and web-based applications. Adversaries can use DoS attacks for political purposes and to support further malicious activities, including extortion, hacktivism, and distraction.

Endpoint Denial of Service denies the availability of a service without drenching the network that can be used to provide access to the service. Attacks target different application stack layers that may reside on the system used to provide the service. These layers can include operating systems (OS), server applications such as, for example, web servers, DNS servers, databases, and (most often web-based) applications that reside on them. Attacking these layers requires different techniques that take advantage of bottlenecks unique to particular components. A DoS attack can be generated with a single system or multiple systems located

throughout the Internet, most often called distributed DoS (DDoS).[9]

In order to perform DoS attacks on endpoint resources, several aspects are applied to multiple methods, including IP address spoofing and botnets.

Attackers can use the original IP address of the attacking system, or they can spoof the original IP address in order to make it difficult to trace the attack traffic to the attacking system or to enable reflection. These methods can increase the difficulty defenders may have in defending against attacks by reducing or eliminating the effectiveness of source address filtering on devices that protect the network.

Botnets are most often used to perform DDoS attacks on networks and services. Large botnets can generate significant amounts of traffic from systems spread across the global Internet. Adversaries may have the resources to build and control their own botnet infrastructure or rent time on an existing botnet to launch an attack. In the worst-case scenario for DDoS, many systems generate requests that each need to send a small amount of traffic in order to produce enough volume to exhaust the target's resources. Under these circumstances, distinguishing DDoS traffic from legitimate clients becomes extremely hard. Botnets were used in some of the incredibly famous DDoS attacks, such as a series of incidents in 2012 that targeted significant banks in America.[9]

In instances where traffic manipulation is used, there may be junctures in the global network (such as high-traffic gateway routers) where packets can be modified. They can also cause legitimate clients to execute code that routes network packets straight to their destination on a large scale. This type of capability has previously been used for web censorship purposes where client HTTP traffic would include a reference to JavaScript that would generate DDoS code to flood targeted web servers.

See Network Denial of Service for attacks attempting to saturate the providing network.[9]

8. Cyber attacks and System Exploits

8.1. Understanding Cyber Attacks

An intentional manipulation of computer systems and networks is referred to as a cyber attack. Cyber attacks are carried out so that attackers can use malicious code to alter computer code, logic, or data, which can lead to disruptions that compromise data and can even lead to cyber crimes, such as information and identity theft. Attackers can use many tools to launch attacks. Some of these tools are malware, ransomware, exploit kits, and various other methods. Some of the most significant attacks are listed later in this chapter.[10]

8.1.1. Malware attack

The consequences that can arise from these attacks can vary from the unauthorized downloading of information to the interruption of certain services. Viruses are programs that connect to other software and then reproduce when that program is launched. Worms are self-replicating malware that scans vulnerable parts of the system and then attacks them. Trojans represent malicious code that can be hidden inside software that is legal and legitimate to avoid detection, thus providing attackers with remote access to infected systems. Trojans primarily spread if an infected file is downloaded from the Internet or via spam messages. **Figure 11** shows how malware can compromise a system. Network traffic was redirected to the hacker instead of the legitimate web server.[10]

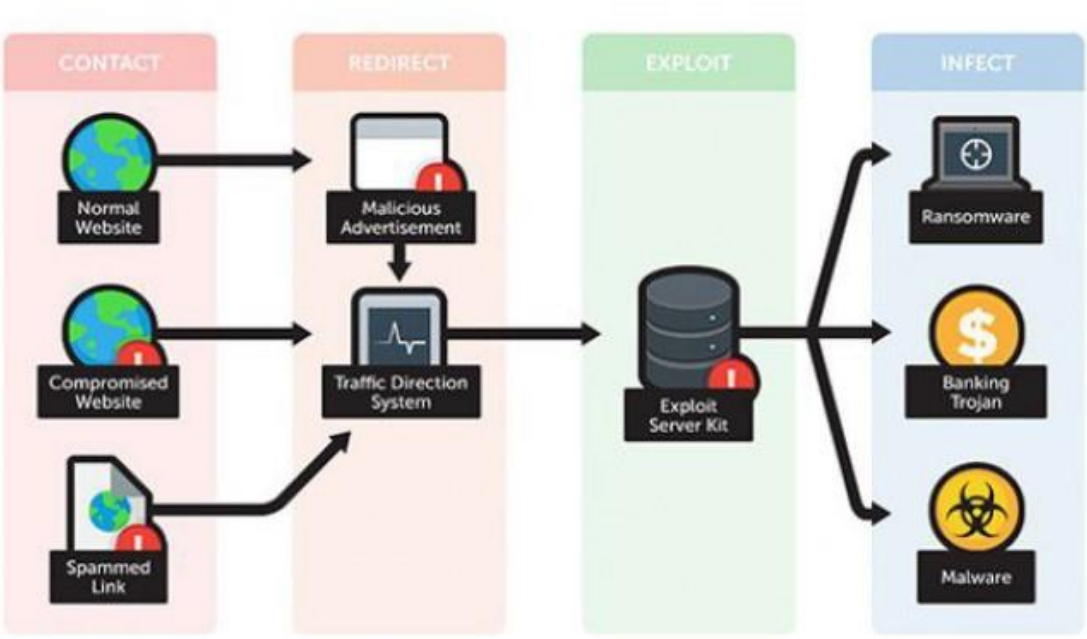

Figure 11: Example of a Malware Attack[10]

8.1.2. Denial-of-Service (DOS)

DOS attacks serve to disrupt the services of a host connecting to the Internet indefinitely. Generally, DOS attacks are carried out by overloading the entire bandwidth of a network or exhausting the resources of the targeted system to the limit. This type of attack is usually carried out via a botnet network. Botnets are networks of already compromised machines infected with some malware and placed under the direct control of an attacker, and they are not active until the attacker assigns them some specific tasks. If the attacker has a large number of botnets, it can cause a distributed denial of service, which can make it much more challenging to locate the source of the attack.[10]

8.1.3. Man-in-the-Middle Attacks

Man-in-the-middle attacks are the most common type of cyber security attack. This type of attack gives attackers the ability to eavesdrop on communications between two targets. The attack takes place between two users who have legitimate communication. giving the attacker the ability to eavesdrop on their conversation.[10]

8.2. System Exploits

An exploit is a set of instructions that can use a flaw or vulnerability in a system to gain unauthorized access or cause unexpected behavior. Scripts or frameworks are mostly used to attack the system targeted by attackers. In most exploits, privilege escalation is applied before the tasks are executed. Data that can be useful in these attacks, as well as scripts, can be found on the Google Hacking Database or on GitHub.

In this example, in order to prove the effectiveness of the tools used to detect intrusions, several test stations were created. The first test station includes the Wazuh tool, which is used to detect intrusions, and the second test station includes the sysdig falco tool for detecting attacks. Both of these crash detection systems use default configurations and/or rules in this example.[10]

The following exploits will be used in the system to test the intrusion detection tools:

1. **Network Scanning** - A worm is malware that is self-contained and self-replicating, thereby infecting other hosts on the network. When spreading, worms usually scan all ports of the infected machine for any open connections and thus infect all connected systems. Such a network scan will be simulated with the help of NMAP (Network Mapper) and the existence of flags will be checked for systems used to detect intrusions. There is an option to run an nginx server using a docker container and run nmap through its ports. Ngink is a name for a web server, which can be used as a proxy, load balancer or HTTP cash.
2. **Denial of Service Attacks** - If the web server receives too many requests from many applications, there is a significant possibility that it will freeze or crash. This type of attack falls under the category of DOS. This type of attack can be simulated by using NMAP to scan all 65K ports available on a web server, repeatedly.
3. **Data Exfiltration** - In Linux, the file can store the actual passwords in an encrypted format. When exploiting, this sensitive file is read, and data is sent via UDP to a specific IP address and a specific port.
4. **File Integrity Exploit** - A folder in Linux does not provide writable capabilities to

   non-root users. In this exploit, an empty file is created and stored in a specific directory.
5. **Unauthorized Command or Anomalous Behavior** - When deploying a web server, access to the web server shell itself should be restricted to other running processes. If a process were to gain access to the shell, it could read sensitive information and break isolation. The exploit in this example will use the docker exec command to launch bash for the ngink container and to check the response of tools used to detect potential intrusions.

6. **Privilege Escalation** - This exploit is one of the most basic and can be used to highlight the integrity of the entire system. This exploit uses setuid (set user ID) to give users the ability to run an executable file, with the permissions they get from the owner of the executable file and to be able to change the behavior in directories.
7. **Malicious Code** - With this exploit, access to the database server is gained, and another program can be launched. This example will use the respawn service to run the "ls" command.[10]

8.3. Exploit Examples

8.3.1. Denial of Service

In this example, a payload is used to attempt to connect to a port on the target system, and if a successful connection is made, the port is closed. The attack is considered unsuccessful if the socket cannot be connected. Even if the connection attempt was unsuccessful, a network footprint is registered on the given target. The process of connecting to the socket can be repeated several times at the same time. Such incessant repetitions lead to network congestion, and eventually to DOS attacks.[10]

8.3.2. Data Exfiltration

This example uses a payload, which has the competence to read the contents of the file and send data to the attacker. In order to achieve this, privilege escalation should be performed before the payload is executed. The attack is executed from the docker0 interface.[10]

9. Sysdig Falco

Sysdig Falco is a behavior-based open-source monitoring tool used to detect suspicious activity. Sysdig Falco can work as an intrusion detection system and is very useful if using **docker** as container-specific contexts like **container id** and **container images** are supported.

Falco has given users the ability to define as detailed rules as possible to check activity, including file and network activity, the ways in which processes are executed, communication between processes, etc. Falco alerts are triggered by specific system calls, their arguments, and the properties of the calling process. Falco runs in user space, using a kernel module that intercepts system calls.

Sysdig's filter expressions are used by Falco rules to identify suspicious activity and can send notifications to files, the system log, or to programs. With containers, Falco runs as a container to monitor the host and all containers running on it. **Figure 12** shows the flow of information at Sysdig Falco.[10]

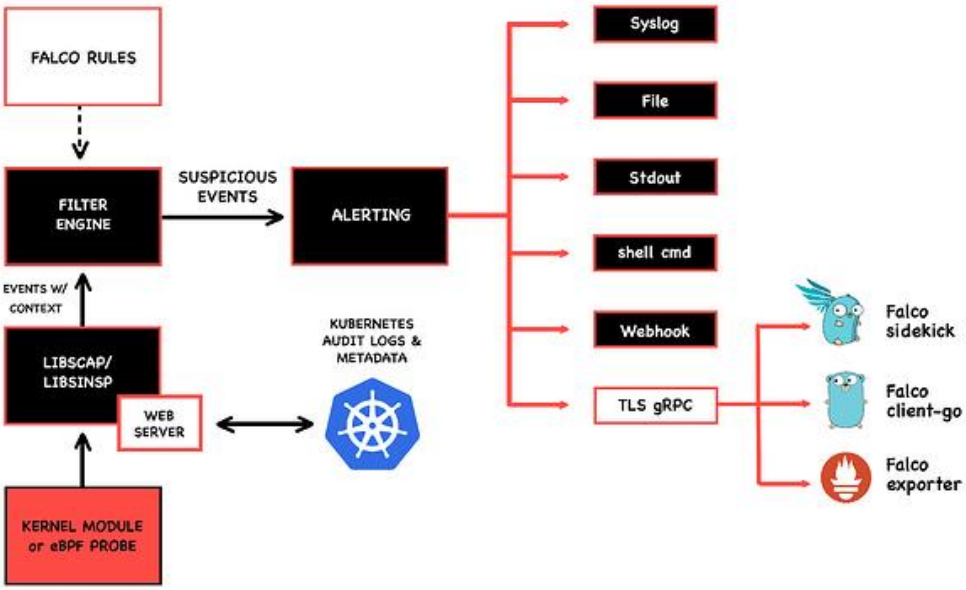

Figure 12: Sysdig Falco Information Flow[10]

Since we covered the basics of Sysdig Falco, let’s take a look at how Sysdig Flaco behaves in Kubernetes clusters since that is the environment in which the implementation is being tested. Falco deployment in the Kubernetes cluster is handled by Helm chart. By taking

care of all the k8s objects required by Falco to be fully integrated into your environment, this chart operates the lifetime of Falco in a cluster.

9.1 Deployment of Falco

The start of Falco set-up begins by choosing whether we deploy it using a deployment or by using a daemonset. As a next step, it is imperative to add a falcosecurity repository before the installation of the Helm chart. As discussed, Flaco can be deployed as a deployment or as a daemonset by the charts, but it all depends on the event sources. In the past, Falco monitored kernel-level system events in an effort to spot malicious activity on Linux computers. It might also analyze Kubernates Audit Logs to look for suspicious activity in Kubernetes clusters. But the Kubernates Audit Logs were removed and made into plugins bringing immense change to Flaco. Making Falco event sources come from system events such as drivers and plugins. And thanks to those very same plugins, Falco plugin system can be it's next evolution.[11]

It is important to understand these system events, so as the next step, I will talk about and explain the purpose of drivers and plugins. Falco requires a driver that intercepts the flow of system calls and sends them to Falco. On the node where Falco is operating, the driver has to be installed. By default, a script that either attempts to create the driver dynamically or downloads a prebuilt driver as a backup manages the drivers using an init container. Typically, nothing has to be done.

Falco is extended to handle other data sources via plugins. The current plugin framework is compatible with plugins that can do the following things:

- capabilities for sourcing events
- capacity for field extraction

Because plugin capabilities may be combined, a single plugin can have both of them. Alternately, we may load two distinct plugins, each with their own set of capabilities, one acting as an extractor and the other as a source of events. The Falcosecurity Json plugins and the Kubernetes Audit Events are two excellent examples of this. We have assistance for the K8s Audit Logs in Falco by the deployment of such plugins. Also while using plugins the driver does not have to be present.

Let's talk about how Falco is installed in Kubernetes after clarifying the various event sources and how Falco consumes them using the drivers and plugins. Depending on the event sources, the chart deploys Falco via a deployment or a a daemonset.[11]

9.1.1 Falco in Kubernetes as a daemonset

Falco is used as the daemonset when the drivers are used. K8s ensures that a Falco

instance will be operating on every one of our nodes even when we add new nodes to our cluster by employing a daemonset. Therefore, it is the ideal solution if we need to monitor every node in our cluster. Falco is deployed with the kernel module using the helm chart's default values.[11]

9.1.2 Falco in Kubernetes as a deployment

We previously saw how falco was deployed as a daemonset. Time to see the ideal strategy to install Falco as a k8s deployment. In the case when plugins are utilized as data sources. There are two different sorts of plugins: those that use the push model and those that use the pull model. The push model being used by a plugin, anticipates getting data from a distant source at a certain endpoint. They only expose an endpoint and wait for data to be submitted; for instance, Kubernetes Audit Events anticipates that the k8s api server will send the data when setup in this manner. However, some plugins that follow the pull paradigm get their data from a specific remote server. Now we will take a look as to why a kubernates deployment is better suited when Falco is deployed with plugins:[11]

- While importing logs directly from distant services, they don’t have to be accessible;
- A one active replica is required; otherwise, many Falco instances will receive and send events;

10. Implementation

The current thesis has attempted to propose several possible authentication methods for improving security systems. It has at the very center and the main focus of the topic, honeytokens.

The structure of the network is set in multiple EKS (Elastic Kubernetes Service) Kubernetes clusters with security groups between each mentioned earlier cluster. It is finally time to put all the words into action and see how honey tokens are used. A collection of machines known as a Kubernetes (K8s) cluster allows for the effective, automated, distributed, and scalable operation of containerized workloads. Engineers may coordinate and monitor containers across several physical, virtual, and cloud servers using K8s clusters. This allows for agile and reliable deployments by separating the containers from the underlying hardware layer. It may be challenging to keep up with all the jargon used in the container industry. Let's take a moment to explore the major elements of a Kubernetes cluster in order to provide a more thorough definition before moving on. To start with let’s take a look at the Kubernetes Cluster's Components

- The control panel makes the enormouse abstraction of Kubernates feasible. It allows for the automatic implementation of the configurations you supply for your cluster. The kube-controller-manager, which controls the cluster, the kube-apiserver, which exposes the K8s API, and the kube-scheduler, which monitors the health of your cluster and schedules the deployment of pods to nodes in line with your configuration, are also included in the control plane.
- Workloads the programs that Kubernetes runs are referred to as workloads. A workload might be made up of a single discrete component or several different components operating simultaneously. A K8s cluster divides a task among a number of pods.
- In a Kubernetes cluster, pods are collections of containers that share network and storage resources. The specifications for the containers are also contained in the pods of a Kubernetes cluster.
- Nodes the actual processing units (CPUs) and memory on which workloads are carried out. Regardless of the real source, which might be a virtual machine, an on-premises physical server, or cloud infrastructure, the "hardware" resources in a K8s cluster are represented by nodes.[12]

The structure does not end there; there is as well, Calico, a sort of Kubernetes firewall. The use of this third-party solution is to furnish a more straightforward configuration of Kubernetes network connectivity and to provide flexibility. Before continuing further, it is

important to mention why we are using EKS Kubernetes. EKS remarkably streamlines Kubernetes deployment on AWS, which makes it a handled CaaS (containers-as-a-service).

The scenario for the thesis's implementation begins with the infrastructure having no possible way of detecting any attacks. It was also challenging to reproduce our adversaries' attacks, considering that we did not even have honeytokens placed nor created in the infrastructure at the time. So running the commands, attackers use found in the MITRE table was impossible. The most crucial step is figuring out which attacks are the priority to protect from and trying to fool the attackers. The final decision has been brought forth to try and prevent four attack techniques: gathering bash_history, gathering fake SSH keys, gathering fake AWS keys, and planting an SSH "backdoor." The reasoning as to why each of these techniques was chosen differs from the techniques themselves.

For bash history, it was imperative to have a fake .bash_history for the publisher user, which will trigger alerts on access. In the case of private key attacks, for the purpose of protection, a publisher would need to have fake SSH credentials and also have high-severity alerts for accessing them, which means getting notified as soon as someone accesses these credentials. For the previous type of attacks, the same could be applied to attacks regarding AWS keys to protect publishers by creating fake AWS credentials and putting high-severity alerts for accessing them. The last attack which was simulated in the current implementation is backdoor SSH access, where the goal is to prevent the attacker from gaining entry into the infrastructure in order to secure remote access to the publisher's system.

The practical work on which this thesis is based consists of the following lines of code that are mainly used by attackers trying to break into the publisher's system. For the .bash_history attacks, the central concept was taken from Metasploit. A project whose purpose was made to help security teams verify vulnerabilities to help them improve their security awareness and to aid them in managing security assessments. In order to ensure that the defenders stay at least a few steps in front of the attackers. Metasploit has shared its idea with the public on "GitHub"[13], where its primary goal is to gather user-specific information such as SQL histories, last logs, etc.

In order to test their attack idea, we first needed to create a fake **.bash_history**, which will have the honeytoken role, to attract attackers into thinking that he is exploiting our weakness. Once the phony .bash_history is made, we will run a small simulation trying to access it from our command prompt using the following line of code shown in **Figure 13**.

```
cat /home/*/.bash_history >> /tmp/gathered_credentials.txt_
```

Figure 13: .bash_history attack code

The function of the code line is that we open, in this case, the defenders/**home** directory searching for any information or credentials we can use. When the attacker has looked around the defender's files and folders, the final destination would be the .bash_history harboring the gathered_credentials.txt file.

Thanks to Sysdig Falco, who was introduced to us in a theoretical manner in the 6th chapter, where it will be used as a critical player for setting the rules of our notifier and logger. The name of the rule chosen to be used in the implementation is "Honeytoken access detected". This rule will be the soul and heart of the notifying purpose of Sysdig Falco, which is set up in **Figure 14**.

```
- rule: Honeytoken access detected
  desc: 'We planted a few files we usually don''t need / use but attackers are typically
looking for. '
```

Figure 14: Rule name

The name of the rule will be present on the main dashboard, as shown in **Figure 15**. We will be setting up conditions which, when they are met, Sysdig Falco will call this rule and notify the defenders.

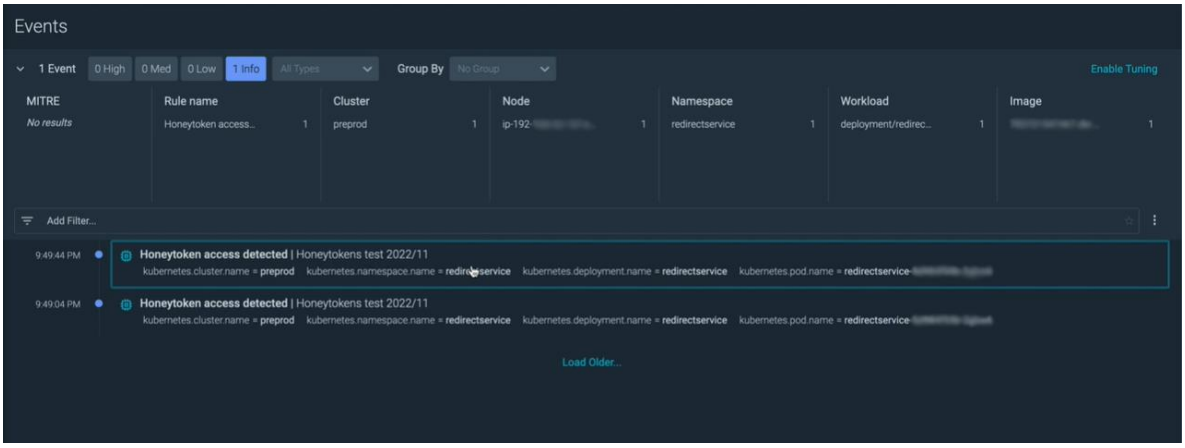

Figure 15: Rule displayed on Sysdig dashboard

Conditions which were used for the "Honeytoken access detected" rule were: if anyone opens to read a file with a specific name that belongs to the honeytoken_read_files list or the opponent decides to edit the file with that particular name in the honeytoken_read_files list. For a better illustration, refer to **Figure 16**.

```
  condition: (open_read and proc_name_exists and fd.name in (honeytoken_read_files)) or
(open_write and proc_name_exists and fd.name in (honeytoken_write_files))
```

Figure 16: Conditions set for the rule

Sysdig Falco will give an output containing helpful information that the defender can use to track how the attacker got in, from which side, and even their IP address. **Figure 17** shows the output that the defender has set.

```
  output: Sensitive file opened for reading by non-trusted program (user.name=%user.name
user.loginuid=%user.loginuid program=%proc.name proc.cmdline=%proc.cmdline file=%fd.name
parent=%proc.pname gparent=%proc.aname[2] ggparent=%proc.aname[3] gggparent=%proc.aname[4]
container.id=%container.id evt.type=%evt.type evt.res=%evt.res proc.pid=%proc.pid
proc.cwd=%proc.cwd proc.ppid=%proc.ppid proc.pcmdline=%proc.pcmdline proc.sid=%proc.sid
proc.exepath=%proc.exepath user.uid=%user.uid user.loginname=%user.loginname
group.gid=%group.gid group.name=%group.name container.name=%container.name
image=%container.image.repository)
```

Figure 17: Output for set rule

In the output, the details of the containers which are being used alongside the clusters. These details are shown in the sidebar that will appear once we click on the notification in the Sysdig Falco. The defenders will know when the attack has occurred and what has happened, which is basically the description of the rule. In our case, we planted a few files we do not need or use, but the attackers are typically attracted to them. Let us take a look at the output after simulating the attack on our fake .bash_history file.

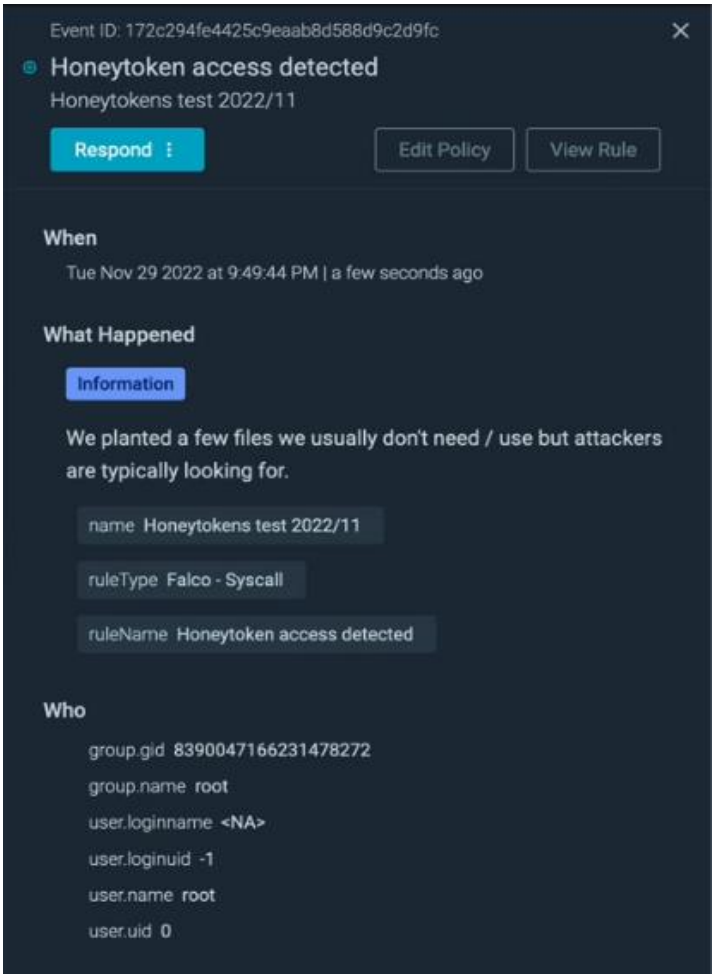

Figure 18: Output of the Attack on .bash_history

Figure 18 shows when the attack occurred with a strict date and time of day, as well as how long ago. What has happened, and who has made the attack. We can see the user's login name and name of the user since it was set in the output with the codes: **user.loginname=%user.loginname** The following **Figure 19** shows us detailed information about the attack, which cluster has the attack affected as well as which container. Just like how the defender has set in the rules with the code **container.name=%container.name,** we can see the container's name in the output as well as its ID, repository, image, and namespace. Meaning that, indeed, as mentioned in the beginning, we can see that the implementation is using the Kubernetes clusters and containers in its network structure.

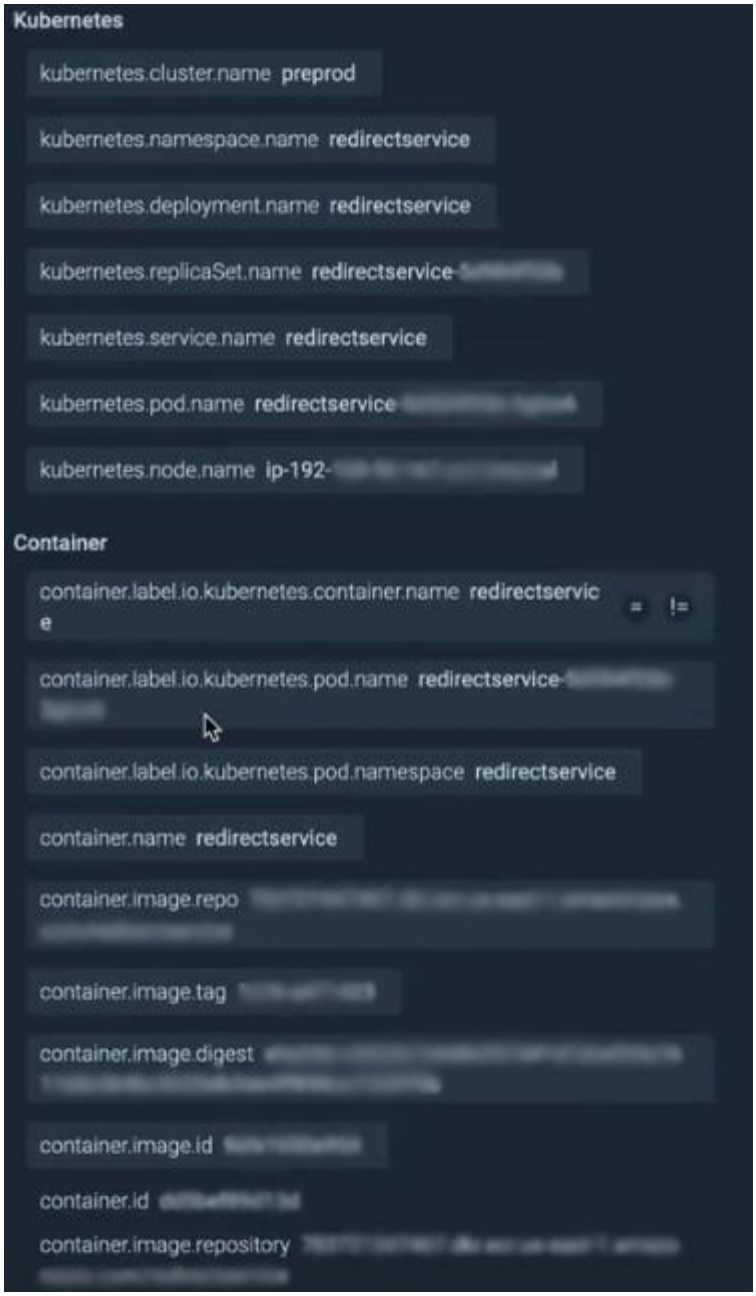

Figure 19: Details from the output

With the help of **Figure 20**, we can see which folder has been accessed and its location; based on that, we can trace how the attacker has gotten to our honeytoken file. However, an even better result is printed in the output under the name **proc.cmdline**. With it, we can see what command line the user used to get to our file. And like at the beginning of the explanation, we finally get the result of what command the attacker used to attack. We can see that **cat**

/home/wwwdata/.bash_history is the same line that was executed in the implementation to simulate the attack.

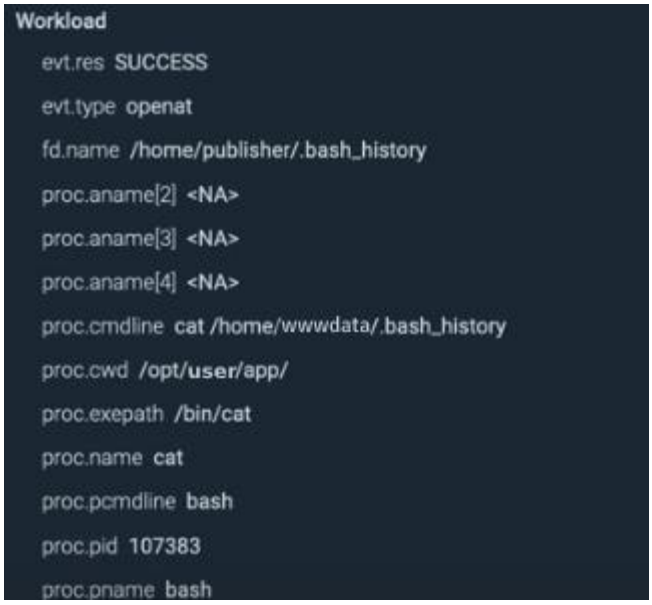

Figure 20: .bash_history attack path

The final information that is printed in the output is the host's information. As shown below in **Figure 21**, the host's name, well, in this case, instead hosts local IP address as well as their physical MAC address as well as the environment that the host is in, and the host's cluster.

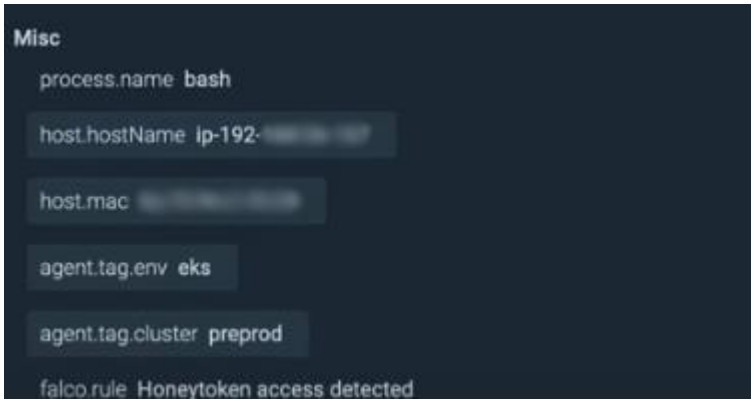

Figure 21: Hosts information

With a thorough explanation of the first attack and the monitoring tool that is being used, the rest of the next attack, which has been simulated, will only contain the command codes and output. It starts with the creation of fake SSH credentials, which will be stored in the /home directory as well as the /root directory. The purpose of this is to monitor how deeply the attacker

has infiltrated the network, as well as testing whether the monitoring tool will notify the defender in case the attacker has gone deeper than just the /home directory. Once the fake credentials are in place, the simulation can begin with the execution of the following lines of code in **Figure 22**.

```
cat /home/*/.ssh/* >> /tmp/gathered_credentials.txt
cat /root/.ssh/* >> /tmp/gathered_credentials.txt_
```

Figure 22: SSH attack code

After the code is executed, the attack should be detected by Sysdig Falco and displayed on the dashboard under the name of the rule set for this implementation, "Honeytoken access detected". Clicking on the rule displayed on the dashboard opens the sidebar that contains all the information set in the output of the rule shown in **Figure 23** and **Figure 24**. Under the workload title, the folder name will be displayed to which the attacker has access.

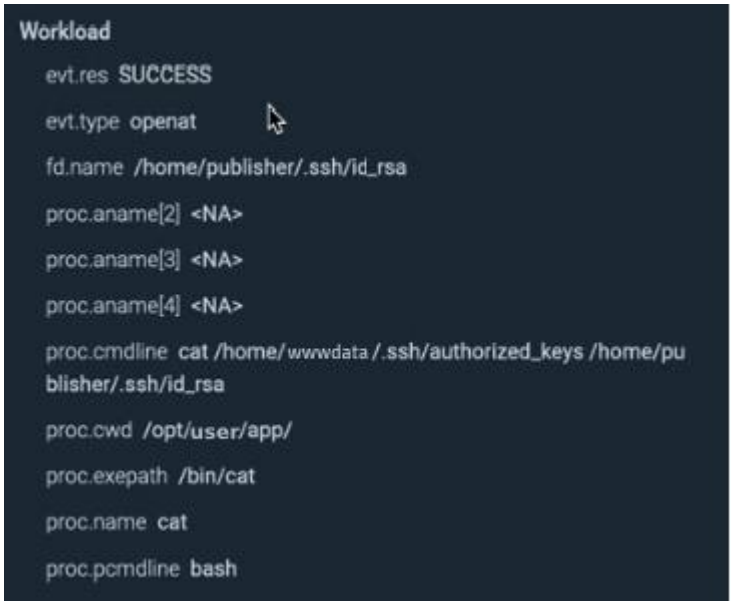

Figure 23: SSH home directory result

With the proof of **Figure 23**, the simulation of an attack on the /home directory was successful, as well as the monitoring displaying which folder has been accessed and using which command line. In**Figure 24** , the visible proof is present, indicating that the /root directory has been accessed by the attacker alongside which command line the attacker used.

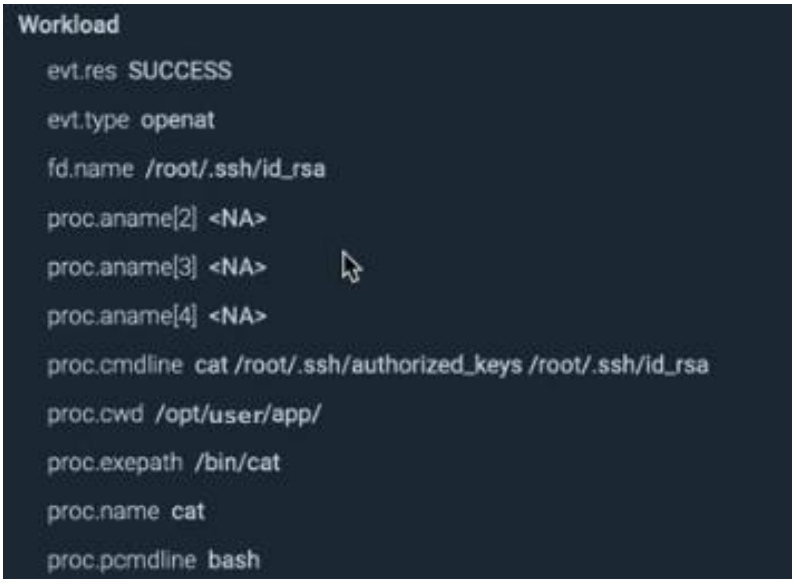

Figure 24: SSH root directory result

The third attack which will be simulated is the creation of fake AWS credentials, which as well will be placed in a /home and /root directory. The following command lines in **Figure 25** will carry out this attack.

```
cat /home/*/.aws/* >> /tmp/gathered_credentials.txt

cat /root/.aws/* >> /tmp/gathered_credentials.txt
```

Figure 25: AWS attack code

The monitoring tool has successfully noticed this attack and is trying to draw the defender's attention by displaying the "Honeytoken access detected" rule on the dashboard. Information printed in the output shown in **Figure 26** lets the defender know the command the attacker has used to gain access to the file, alongside the path of the credential files.

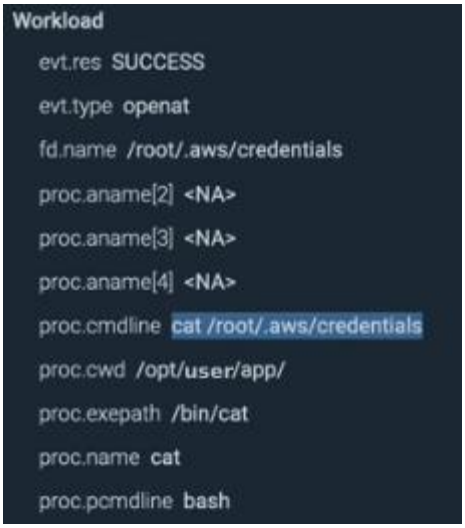

Figure 26: AWS monitoring results

The final attack simulated for testing purposes to help complete and gather data for the current thesis is a backdoor SSH attack using the command codes from **Figure 27**.

Figure 27: Backdoor SSH code

Last recorded and collected data of the attack prove and show that with the monitoring implementation functions, we are able to see the path of the last fake credentials file. With **Figure 28**, the results are displayed.

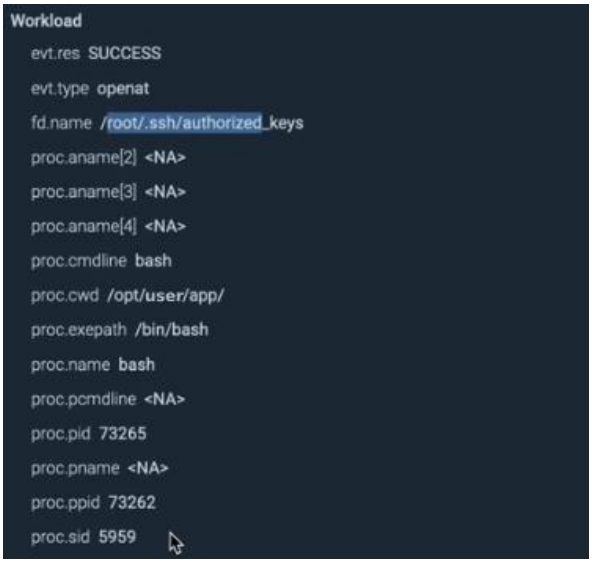

Figure 28: Backdoor SSH monitoring results

11. Conclustion

Seeing honeytokens in action with a massive theoretical background has brought us to a very important conclusion. That the ability of the defender to quickly react to an attack or some incident can aid in minimizing the damage which has been caused by the breach if the defender has a very dependable notifying system that will warn him on time that some malicious behavior is happening, it means that they have a strong incident response plan. Now once a breach or an attack occurs, it is very imperative for us as a defender to make a decision, do we replace and destroy the breached container or try and use it to our advantage to segregate and study the breached container? The purpose of the thesis is to use honeytokens and collect as much information as possible from the attacks. So segregating the breached container and investigating it would definitely be a better choice. If the defender has the experience and has prepared countermeasures in case something goes wrong. Let's take a quick look, how we can segregate and investigate the breached container. Firstly we need to locate where the attack occurred and segregate the Pod along with its nodes from the infrastructure.[14]

There are three ways to identify the breached pods, such as:

- Using the workloads name
- With the name of the service accounts
- Using the vulnerable images

Once the node is identified, the next course of action should be to isolate it. We can do it by creating a network policy that will turn down any ongoing traffic entering and leaving the Pod. Another safety measure before we start investigating the attack is to retract any security credentials which were covering the Pod or any of the Pod's worker nodes. The breacher may try to cover up his tracks by destroying one of the damaged nodes, so there is a method to try and prevent it. We can do it by enabling termination protection. Finally, once all safety measures are in place, the Pod and nodes should be labeled as an active investigation to warn anyone not to tinker with the pods until completion.[14] In the **Figure 29** below, a paid version's UI allows a way to automatically kill/pause containers if an alert is hit.

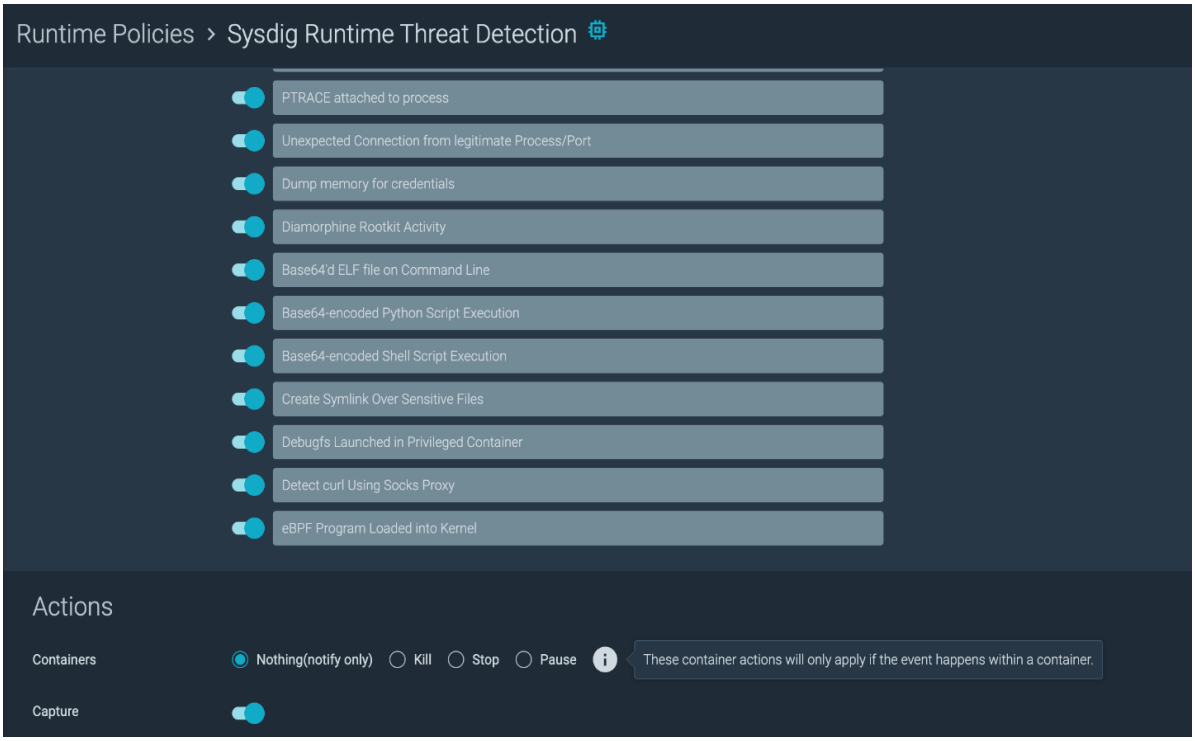

Figure 29: UI actions deployed

For future work, this implementation of information gathering will allow security teams to prevent such issues from happening again. Starting by fixing the entry points or by discovering any new ways.

12. Summary

Developers and security researchers are in charge of securing the Internet in the future. By enhancing security generally, they will raise the bar or make it more difficult for hostile individuals to compromise distant systems. In this work, we've looked at a variety of honeytoken-based computer security solutions that were used to spot, block, or otherwise thwart attempts at unwanted access to databases and information systems.

Hopefully, the purpose of the honeytoken implementation might be utilized in addition to firewalls, web application firewalls, and other security measures to provide extra protection against the dangers outlined in this thesis. Like numerous platforms for filtering, raising the bar, and increasing its resistance to invasions. As we have seen in the practical part, we have successfully simulated attacks, and we have seen the outcome that the honeytokens have had on such attacks as well as which information the honeytokens provided for our research purpose. Our aim was to design, generate and place honeytokens into the system in order to detect and alert attacker’s actions. Using the monitoring tool Sysdig Falco with preset rules, the result has shown a lot of valuable information regarding the attacks.

In comparison to utilizing merely the conventional technique (firewall/IDS/WAF), the suggested solution would allow for more effective identification, containment, and deterrence of automated tools and hackers. Since the firewall won't be able to detect critical components of the program that one wishes to protect and prevent parameter manipulation.

After all, obviously, this does not imply that the other safeguards are no longer required; rather, it is an extra safeguard with the straightforward goal of making the web application more resistant to hacking efforts.

13. References

[1] “Honey Tokens: What are they and How are they used?,” Fortinet. https://www.fortinet.com/resources/cyberglossary/honey-tokens (Accessed Dec. 08, 2022).

[2] J. K. August 15 and 2019, “Making the Most of an Investment in Deception Technology.” https://www.bankinfosecurity.com/making-most-investment-in-deception-technology-a-12881 (Accessed Dec. 09, 2022).

[3] “You Can Catch More Hackers With Honeypots Than Honey.,” https://microage.ca/nwd/en/. https://microage.ca/nwd/en/you-can-catch-more-hackers-with-honeypots-than-honey/ (Accessed Dec. 09, 2022).

[4] V. Papaspirou, L. Maglaras, M. A. Ferrag, I. Kantzavelou, H. Janicke, and C. Douligeris, “A novel Two-Factor HoneyToken Authentication Mechanism.” arXiv, Jan. 20, 2021. doi: 10.48550/arXiv.2012.08782.

[5] G. Bae, H. Y. Lee, and D. It, “Poster: Web Beacon Based Detection of Data Leakage for Android,” p. 2.

[6] “Demystifying KMS keys operations, bring your own key (BYOK), custom key store, and ciphertext portability | AWS Security Blog,” Mar. 12, 2021. https://aws.amazon.com/blogs/security/demystifying-kms-keys-operations-bring-your-own-key-byok-custom-key-store-and-ciphertext-portability/ (Accessed Dec. 09, 2022).

[7] M. Bercovitch, M. Renford, L. Hasson, A. Shabtai, L. Rokach, and Y. Elovici, “HoneyGen: An automated honeytokens generator,” in Proceedings of 2011 IEEE International Conference on Intelligence and Security Informatics, Jul. 2011, pp. 131–136. doi: 10.1109/ISI.2011.5984063.

[8] M. K. Asif and Y. S. Al-Harthi, “Intrusion detection system using Honey Token based Encrypted Pointers to mitigate cyber threats for critical infrastructure networks,” in 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Oct. 2014, pp. 1266–1270. doi: 10.1109/SMC.2014.6974088.

[9] “MITRE ATT&CK®.” https://attack.mitre.org/ (Accessed Dec. 09, 2022).

[10] S. Sridhar, “TESTBED DESIGN FOR EVALUATION OF ACTIVE CYBER DEFENSE SYSTEMS,” p. 57.

[11] “charts/falco at master · falcosecurity/charts,” GitHub. https://github.com/falcosecurity/charts (Accessed Dec. 09, 2022).

[12] “What is a Kubernetes Cluster?,” Check Point Software. https://www.checkpoint.com/cyber-hub/cloud-security/what-is-kubernetes/what-is-a-kubernetes-cluster/ (Accessed Dec. 09, 2022).

[13] “Metasploit.” Rapid7, Dec. 08, 2022. Accessed: Dec. 09, 2022. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/9b62242974e956f689a6aaf8ac9d540582e04d64/modules/post/linux/gather/en

um_users_history.rb
[14] “Incident Response and Forensics - EKS Best Practices Guides.” https://aws.github.io/aws-eks-best-practices/security/docs/incidents/ (Accessed Dec. 09, 2022).

14. Table of Figures