Malware Analysis and Prevention Strategy

Index

Topic	Page
Introduction	3
Chapter 1: Malware Analysis 1.1Understanding Malware: Type and Classifications 1.2 Malware's Circulation 1.3 Malware's Infection 1.4 Malware's Concealment 1.5 Malware's Payload Capabilities 1.6Malware's Dynamic Analysis vs. Static Analysis 1.7 Malware's Real-life examples	5

Introduction

In today’s digital landscape, the prevalence and sophistication of malware pose significant threats to organizations and individuals alike. As cybercriminals continuously evolve their tactics, it becomes imperative for security professionals to stay ahead by understanding the various types of malware and their potential impacts. This project, titled "Malware Analysis and Prevention Strategy," aims to provide a comprehensive framework for analyzing different malware types, implementing effective monitoring solutions, and developing proactive prevention strategies.

Over the course of four weeks, this project will focus on in-depth malware analysis, leveraging Security Information and Event Management (SIEM) systems for real-time monitoring, and creating robust user awareness training materials. The first week will concentrate on researching and documenting the characteristics and impacts of malware, forming the foundation for our subsequent efforts. In the second week, we will configure a SIEM system to enable effective monitoring and alerting for malware activities, ensuring timely detection

and response. The third week will be dedicated to developing a comprehensive malware prevention strategy, coupled with user training materials to enhance awareness and preparedness among users. Finally, the project will culminate in a final report and presentation that synthesizes our findings and recommendations, aimed at fostering a resilient cybersecurity posture. By undertaking this project, we aspire to equip organizations with the knowledge and tools necessary to combat the ever-evolving threat landscape, ultimately contributing to a safer digital environment.

Chapter 1: Malware Analysis

1.1 Malware Analysis: Type and Classifications

Malware, or malicious software, refers to any software that infiltrates a computer system without the user's knowledge or consent, subsequently performing harmful and unwanted actions. The term encompasses a wide variety of damaging software programs. As security measures have advanced to combat malware, the complexity of these threats has also increased, leading to the emergence of numerous distinct malware variants, such as the ZeuS malware. However, there is no universally accepted classification system for these various instances, and many existing classifications simply list different types of malwares (e.g., viruses) rather than grouping similar instances into broader categories. This lack of standardization can make understanding malware classifications confusing.

One effective way to categorize malware is by focusing on its primary traits: circulation, infection, concealment, and payload capabilities.

1. **Circulation**: Some malware is primarily designed to spread quickly to other systems, impacting many users. This circulation can occur through various means, such as network connections, USB flash drives shared among users, or email attachments. Depending on the malware, it may propagate automatically or require user action to spread.

2. **Infection**: Once malware has circulated to a system, it must "infect" or embed itself within that environment. This can happen either through a one-time execution or by remaining on the system for repeated activation. Some malware attaches itself to benign applications, while others operate as standalone processes.
3. **Concealment**: Certain types of malwares prioritize evading detection by software scanners that search for malicious programs. They may do this by altering their code or embedding themselves within legitimate processes, or by modifying the underlying host operating system.

4. **Payload Capabilities**: When the primary trait of malware revolves around its payload capabilities, the focus shifts to the harmful actions it performs. This could include stealing passwords and sensitive data, deleting essential programs, or altering system security settings. In some cases, the malware aims to use the compromised system to launch attacks on other computers.

And in the Next Sections we will talk about deeply in different types of malwares according to these classifications.

1.2 ***Malware's Circulation***

Two types of malware have the primary trait of circulation. These are viruses and worms.

1.2.1 Virus

A biological virus is an agent that reproduces within a host cell, commandeering the cell's functions to produce numerous copies of itself. Once a cell is infected, it rapidly generates thousands or even millions of identical virus copies—like the polio virus, which can create over a million replicas within a single cell. Biologists often assert that the primary purpose of viruses is to replicate themselves. Similarly, a computer virus is malicious code that self-replicates on the same machine. In strict terms, a computer virus can reproduce itself or a modified version without any human assistance.

> ***Note***
>
> Strictly speaking, virus and malware are not interchangeable terms. A virus is only one type of malware

Viruses typically infect computers by embedding themselves in files, which can be either executable programs or user-created data files. A virus that attaches to an executable file is known as *a program virus* and activates when the program runs. Viruses can also reside within data files, with macro viruses being among the most common. A macro is a set of instructions combined into a single command, often used to automate complex or repetitive tasks. These macros can be created using scripting languages like Visual Basic for Applications (VBA) and are stored in user documents, such as Excel (.xlsx) or Word (.docx) files. When the document is opened, the macro executes, regardless of whether the instructions are harmless or part of a malicious macro virus.

Note

The first macro virus emerged in 1995, primarily targeting Microsoft Word documents and becoming the most common type of virus until 2000, when Microsoft disabled macros by default in its Office products. Despite this, macro viruses have seen a resurgence as threat actors find new methods to deceive victims into enabling macros, allowing these viruses to execute.

Early viruses had a simple infection method known as appender infection. In this approach, the virus attaches itself to the end of the infected file and adds a jump instruction at the beginning that directs execution to the end of the file, where the virus code starts. When the program is run, the jump instruction takes control to the virus. Figure 1-1 illustrates how an appender infection operates.

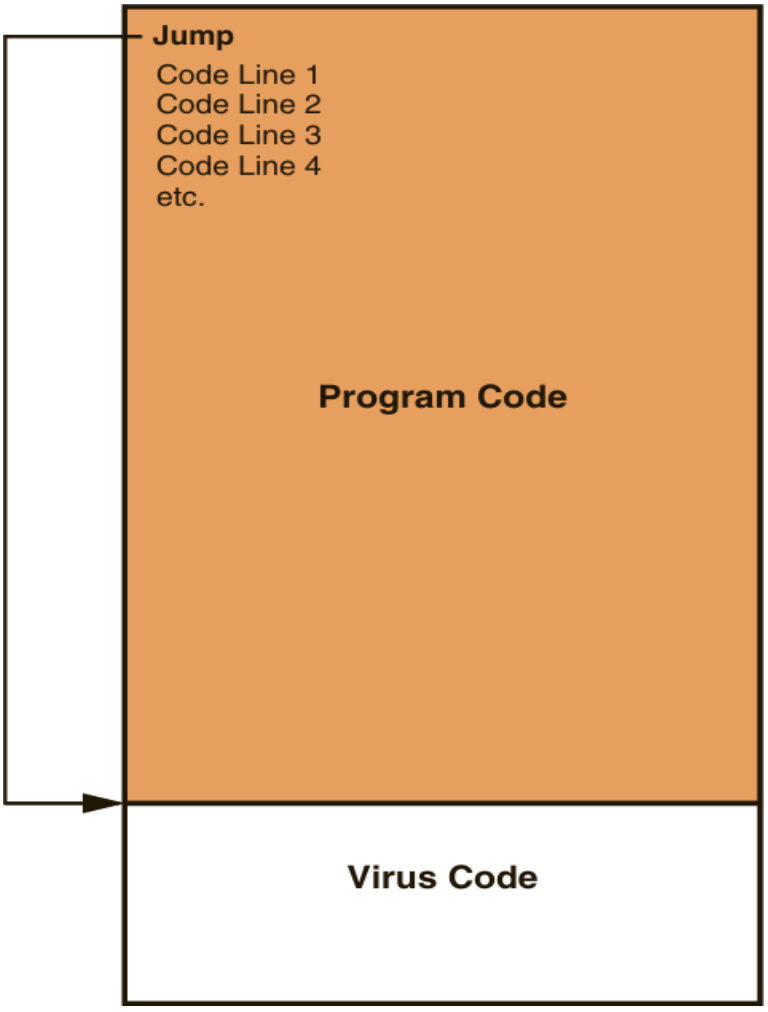


Fig [1-1] Describes Appender Infection

However, these types of viruses could be detected by virus scanners relatively easily. Most viruses today go to great lengths to avoid detection; this type of virus is called an armored virus. Some of the armored virus infection techniques include:

Swiss cheese infection: which involves a dual-layered approach. Instead of a single jump instruction leading to the main virus code, these viruses first encrypt their malicious code, rendering it more challenging for scanners to identify. They then fragment the decryption engine into multiple components, scattering these pieces.

throughout the infected program's code. Upon execution, the program reassembles these fragments to reveal the original virus, creating a convoluted path for detection, **Swiss cheese infection** derives its name from the resemblance to Swiss cheese, which is characterized by numerous holes and gaps. In this context, the "holes" refer to the way the armored virus splits and distributes its code throughout the host program. Just as Swiss cheese has irregularly placed holes, the virus fragments are scattered at various points within the code, making it difficult for antivirus software to detect the malicious components. A Swiss cheese infection is shown in Figure 1-2.

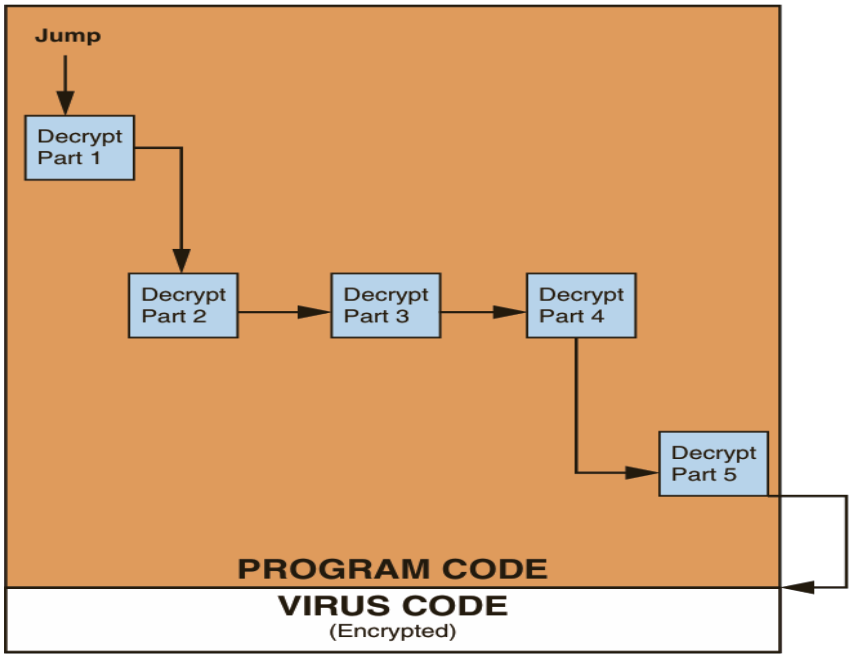


Fig [1-2] Describes Swiss Cheese Infection

split infection In this method the virus itself is divided into several segments, which are interspersed randomly within the host program's code. To further obscure its presence, these segments may include extraneous "garbage" code, further disguising the virus's true intent. This strategic placement not only complicates detection efforts but also heightens the risk of infection, A split infection virus is shown in Figure 1-3.

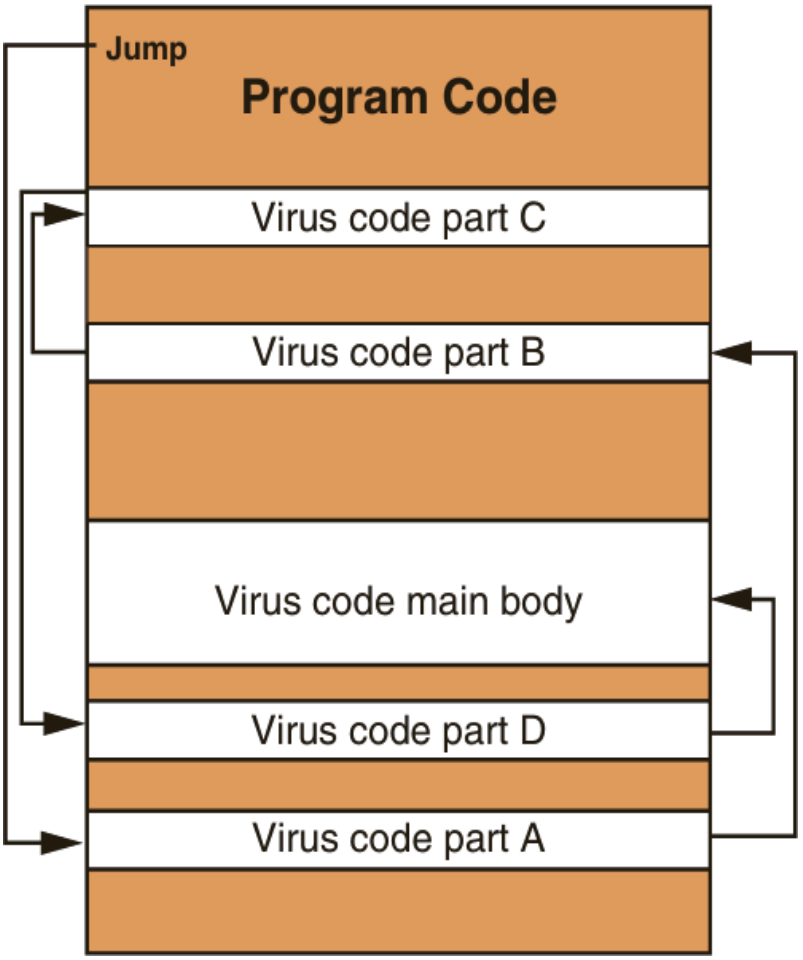


Fig [1-3] Describes Split Infection

Mutation is a critical characteristic of certain computer viruses, allowing them to evade detection and enhance their survival. Unlike simple viruses that merely conceal themselves within files, some possess the ability to alter their code dynamically. **Oligomorphic viruses** modify their internal structure to one of a limited set of predefined mutations upon execution, whereas **polymorphic viruses** entirely change their form each time they are activated. The most advanced, **metamorphic viruses**, can rewrite their own code, generating a logical equivalent that appears different with every execution. And figure 1-4 show the difference between polymorphic and Metamorphic.

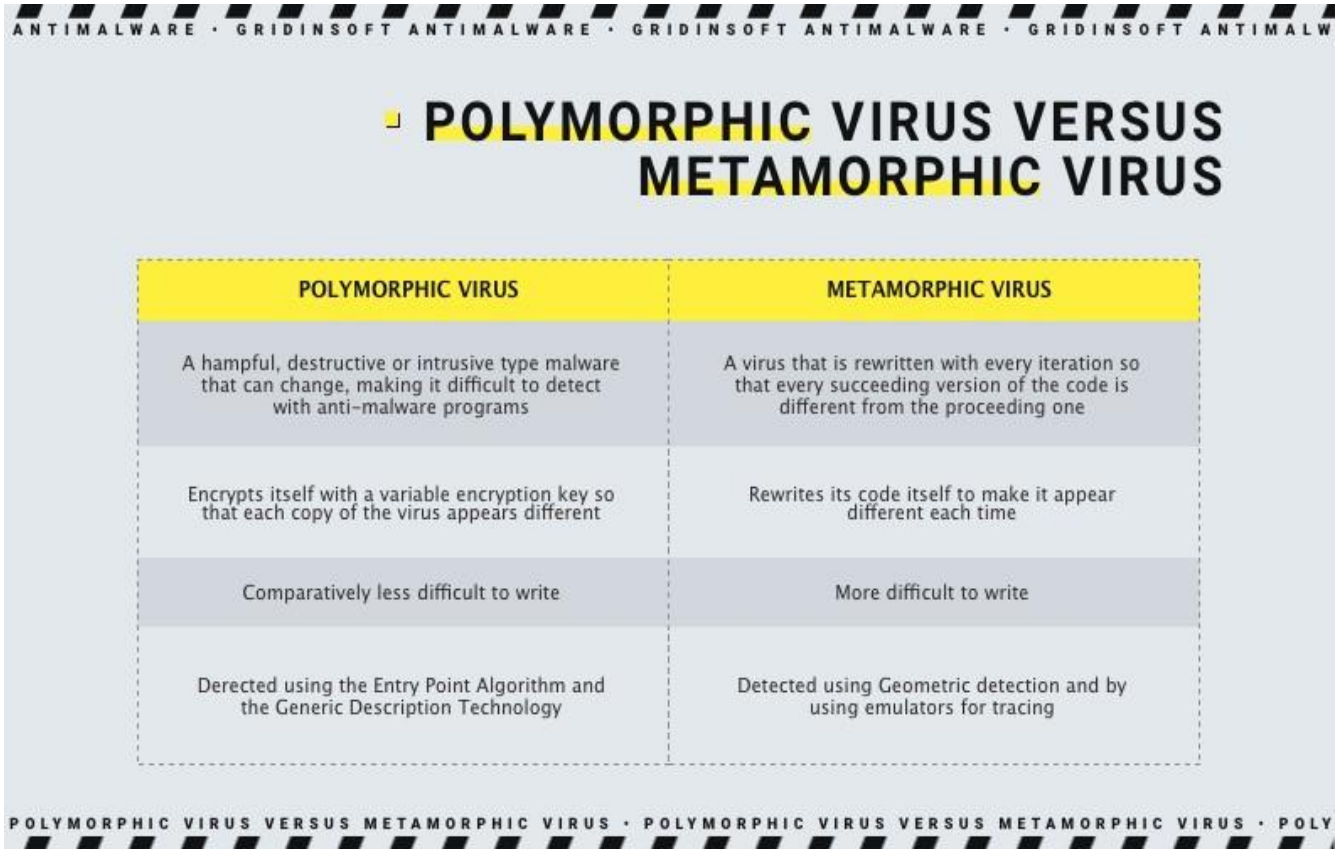


Fig [1-4] Describes the difference between Polymorphic and Metamorphic virus

At the end, when an infected program is initiated or a compromised file is accessed, the virus executes two primary functions. Firstly, it releases a payload designed to execute malicious actions. Modern viruses are significantly more destructive than their early counterparts, which primarily displayed nuisance messages; contemporary variants can corrupt or delete files, prevent applications from launching, exfiltrate sensitive data, instigate repeated system crashes, and disable security settings.
Secondly, upon execution, the virus replicates itself by embedding its code into additional files, but this replication is confined to the original host computer. Viruses lack the capability for autonomous transmission to other systems; they depend on user actions for dissemination. For instance, a user may inadvertently share an infected file via email or transfer it using a USB flash drive. Upon reaching a new host computer, the virus begins the infection cycle anew. Thus, a virus requires two essential components for propagation: a file to which it attaches and a human agent to facilitate its transfer to other systems.

Note

Sometimes a virus will remain dormant for a period before unleashing its payload

1.2.2 Worm

A second category of malware specifically designed for propagation is the **worm**. Worms are malicious programs that exploit computer networks to replicate themselves, often referred to as network viruses. These programs are engineered to infiltrate a computer via the network, taking advantage of vulnerabilities in applications or operating systems on the host machine. Once a worm successfully exploits a vulnerability on one system, it actively seeks out other computers on the network with the same weaknesses.

In their early iterations, worms were relatively harmless, designed primarily for rapid dissemination without causing damage to the infected systems. However, they could significantly degrade network performance, as their rapid replication consumed available resources. Modern worms, on the other hand, often deliver a harmful payload to the systems they infect, akin to traditional viruses. Their actions can include deleting files on the infected computer or enabling remote control by an attacker, thereby posing a considerable threat to both individual systems and network integrity.

Note

Although viruses and worms are said to be automatically self-replicating, where they replicate is different. A virus self-replicates on the host computer but does not spread to other computers by itself. A worm self-replicates between computers (from one computer to another)

Action	Virus	Worm
What does it do?	Inserts malicious code into a program or data file.	Exploits a vulnerability in an application or operating system
How does it spread to other computers?	User transfers infected files to other devices	Uses a network to travel from one computer to another
Does it infect a file?	Yes	No
Does there need to be user action for it to spread?	Yes	No

1.3 Malware's Infection

1.3.1 Trojans

Named after the legendary Trojan horse from Greek mythology, which allowed Greek soldiers to secretly infiltrate the city of Troy, a computer Trojan operates in a similar deceptive manner. It disguises itself as a legitimate application, fooling users into installing it under the belief that it will perform harmless or useful tasks. However, alongside the advertised function, it secretly executes malicious activities, such as:

- Collecting sensitive information like credit card details and passwords.
- Transmitting this information to attackers via remote servers.

A particularly dangerous subtype of Trojan is the Remote Access Trojan (RAT). In addition to its standard Trojan functionality, a RAT allows attackers unauthorized control over the infected device. Once inside, the threat actor can monitor the user's actions, access files, modify settings, and even use the infected computer to spread the infection across a network.

1.3.2 Ransomware

Ransomware is an aggressive form of malware that blocks access to a device or its files until a ransom is paid. After embedding itself into the system, it locks down functions, often launching immediately upon restart to prevent circumvention.
Early Ransomware (Blocker Ransomware): Initially, ransomware focused on freezing the entire computer, displaying a full-screen message that impersonated a legitimate entity, such as law enforcement or software providers. These warnings claimed that the user had committed illegal activities or that critical system issues needed resolution, prompting immediate payment of a "fine" or fee to restore access.
Modern ransomware has abandoned pretense. Now, it simply locks the system, demanding payment for its release. This model has become so lucrative that

ransomware attacks have skyrocketed, with billions of dollars lost each year. Notably, even if victims pay, there's only a limited chance they will successfully recover their files. As you can see in figure 1-5 and figure 1-6 the shape of ransomware infection.

Fig [1-5] and Fig [1-6] Shows the shape of ransomware infection.

1.3.3 Crypto-malware

Building on the foundations of ransomware, crypto malware takes the attack a step further. Instead of just blocking access to a system, it encrypts critical files so that they cannot be opened or used without an encryption key. Victims are told that they must pay for this decryption key, with the threat of permanent data loss if they refuse.

The encryption process begins with the malware connecting to the attacker's Command & Control (C&C) server, where it receives an encryption key. Once the files are locked, the malware sends the decryption key back to the C&C server, and victims can only retrieve it after paying the ransom, and you can see its shape in figure 1-7.

Recent Advances in Crypto malware:

- It now targets not just the local computer but any connected devices and network storage, which could affect entire enterprises.
- Some forms also infect mobile devices, expanding the scope of potential damage

.

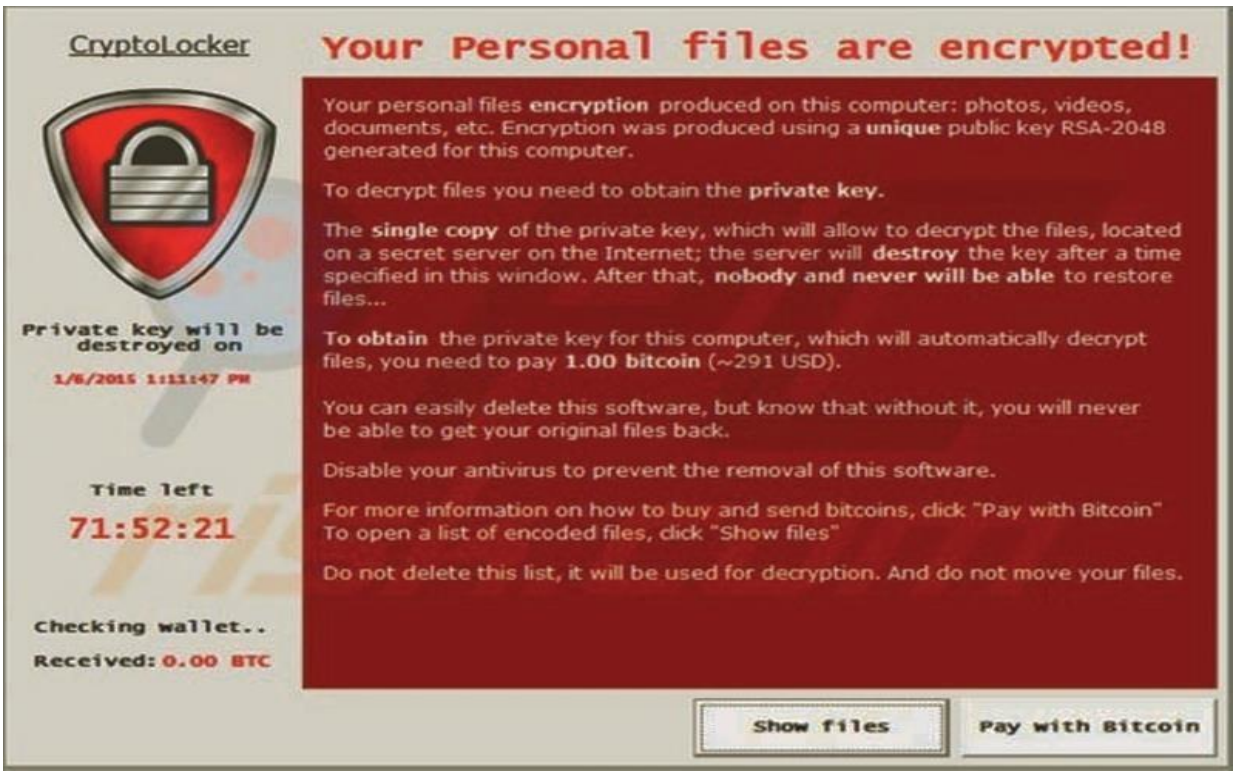


Fig [1-7] Shows crypto-malware infection

Feature	Trojans	Ransomware	Crypto-Malware
Primary Trait	Deception (masquerades as legitimate apps)	Locks system access	Encrypts files, making them unusable
Malicious Activity	Steals data (passwords, credit cards)	Blocks device until ransom is paid	Encrypts files; demands ransom for decryption
Method of Infection	User unknowingly installs malicious software	Embeds into system, launches on boot	Encrypts files after receiving key from C&C
Severity	Medium: Can be detected and removed	High: Prevents system use until payment	Critical: Loss of data unless ransom is paid
Subtypes	Remote Access Trojan (RAT)	Blocker ransomware	Advances include file & network encryption
Impact on Networks	Can spread to other devices in network	Typically isolated to one device	Can infect multiple devices & network storage
Target Devices	Computers and networks	Computers and mobile devices	Computers, networks, cloud storage, mobile devices

Each of these malware types poses significant risks, with Trojans focusing on stealth and data theft, ransomware on extortion, and crypto-malware presenting the most destructive potential through encryption. By understanding their differences, users can take appropriate steps to protect their systems.

1.3 Malware's Concealment

1.4.1 Definition of Concealment

In the field of malware security, concealment refers to the deliberate act of hiding malicious code or activities from detection by security tools and system users. Malware developers use various techniques to ensure that their software operates in the background without raising alarms, such as obfuscating code, embedding malicious components in legitimate files, or using rootkits to mask processes. This concealment is intentional, designed to prolong the malware's presence and effectiveness on a compromised system, often to steal sensitive data or cause damage without immediate detection. Unintentional concealment can also occur when malware exploits unknown vulnerabilities, allowing it to remain hidden for extended periods, sometimes only discovered after significant harm has been done.

1.4.2 Types of Concealment

- Physical Concealment: can be seen in hardware-based attacks where malicious devices or implants are hidden in physical environments (e.g., USB sticks with malware).

- Informational Concealment: is more prominent, as malware often hides critical information from security systems. Techniques such as obfuscation, encryption, and fileless malware allow attackers to keep malicious code and data hidden from antivirus programs and intrusion detection systems.

- Emotional Concealment: can be compared to the way phishing attacks exploit human psychology by hiding the true intent of malicious communications, creating a false sense of trust or urgency, thus leading users to reveal sensitive data or install harmful software without realizing the threat. These layers of concealment make it challenging for cybersecurity professionals to identify and mitigate risks effectively.

1.4.3 Concealment in Law

• Fraudulent Concealment: refers to deliberate actions taken by malicious software, such as rootkits, to hide its presence or the presence of other malware on a system. Rootkits are specifically designed to access lower layers of the operating system, often using undocumented functions, to modify system behavior and evade detection. This manipulation allows rootkits to conceal files, processes, and activities from antivirus programs and operating systems, similar to how fraudulent concealment in law involves intentionally hiding facts to deceive others.

• Non-disclosure: in the malware field can be compared to the failure of certain malware to reveal its harmful payload, allowing it to operate without alerting users or security systems. These tactics not only deceive users but also undermine system integrity.

• Legal Consequences: of malware-related concealment can be severe, leading to fines or imprisonment for those involved in the creation and dissemination of such malicious software. Just as in legal cases, where withholding critical information leads to penalties, malware authors face harsh penalties when their deceit is uncovered, particularly when the malware causes substantial damage to systems or sensitive data.

1.4.4 Concealment in Technology (Cybersecurity)

• Data Concealment: is a common technique, where sensitive information is hidden through encryption or obfuscation, preventing unauthorized access. Malicious actors may use encryption not only to protect stolen data but also to conceal the malware's activities.

• Network Concealment: involves techniques like VPNs, proxy servers, and Tor to hide or mask network traffic, making it difficult for cybersecurity tools to detect or trace back malicious activity.

• Stealth Malware: such as rootkits, epitomizes advanced concealment by embedding itself deep within a system's operating layers. Rootkits can alter the operating system's functioning, making malicious files invisible to security tools and scanners. For instance, when a rootkit infects a computer, it hides both its presence and the presence of other malware by accessing lower layers of the operating system, making these files undetectable by antivirus software.

1.4.5 Military Concealment

Camouflage : can be likened to the stealthy techniques used by malware to hide its presence and evade detection, similar to how armies use camouflage to make personnel, equipment, or positions less visible on the battlefield. Malware, like rootkits, employs advanced "camouflage" by embedding itself deep within a computer's operating system, making malicious files or processes invisible to security tools.

• Deception Tactics: Just as soldiers use deception tactics such as decoys or false information to mislead the enemy, malware employs strategies like misleading system logs, altering file timestamps, or hiding in legitimate processes to deceive antivirus software and users. For example, rootkits access lower layers of the operating system to conceal their presence and that of other malware, making it impossible for scanning software to detect them. This "military-style" concealment allows the malware to continue operating in the background, much like a hidden army quietly advancing, undetected by its adversaries.

1.5 Malware's Payload Capabilities

1.5.1 What does "payload" means in cybersecurity, software, and networking?

In cybersecurity, "payload" refers to the malicious code or actions that are delivered and executed after an exploit or vulnerability is triggered. This can include malware, ransomware, or other harmful programs designed to damage, steal, or manipulate data. In software, the term "payload" often refers to the portion of a program or message that performs the intended function, as opposed to the surrounding code or metadata. In networking, "payload" is the data being transmitted within a packet, excluding the headers and metadata used for routing and transmission. Essentially, it's the core information or executable part that fulfills the purpose of a cyber operation or communication.

1.5.2 What does "payload" means in cybersecurity, software, and networking?

In software, a payload refers to the portion of data or code that delivers the intended functionality, such as a piece of executable code in malware or the main data carried within a network packet. It's abstract and operates within the digital environment, typically involving tasks like data manipulation, communication, or malicious actions in cyberattacks. In physical systems like drones or satellites, however, a payload refers to the physical components or cargo carried by the system. This could be scientific instruments, cameras, sensors, or other equipment designed to accomplish specific tasks (e.g., capturing images, gathering data, or performing experiments). While software payloads are virtual and focused on digital operations, payloads in physical systems are tangible and centered around completing real-world objectives.

1.5.3 Cybersecurity Payloads

- **How are payloads used in cyberattacks (malware, ransomware, exploits)?**

  In cyberattacks, payloads are the core malicious component delivered after an attacker exploits a vulnerability or leverages social engineering techniques. Once the system is compromised, the payload executes, performing harmful actions like encrypting files (ransomware), corrupting systems, or stealing sensitive data (malware). For example, ransomware payloads typically encrypt files and demand a ransom in exchange for a decryption key, while malware payloads might install a backdoor that allows attackers continuous access to the system. Exploit payloads work by taking advantage of system vulnerabilities—such as an unpatched software flaw—to deliver and execute the attacker's desired code. These payloads are crucial for achieving the attacker's ultimate goals, whether they involve financial gain, data theft, or sabotage.

- **Role of payload in a cyberattack lifecycle (e.g., delivery, execution, impact).**

  In the lifecycle of a cyberattack, the payload plays a central role in the delivery, execution, and impact stages. First, the attacker identifies a vulnerability or uses phishing tactics to deliver the payload. This delivery can happen through email attachments, malicious websites, or direct network access. Once delivered, the payload is triggered to execute its intended function, such as installing spyware, exfiltrating data, or disabling system defenses. In the impact stage, the payload fulfills the attacker's objectives, whether by encrypting files in a ransomware attack, establishing a backdoor for persistent access, or launching a DDoS attack. The payload's efficiency and ability to remain undetected determine the overall success and severity of the cyberattack.

- **Common types of malicious payloads (backdoors, keyloggers, data exfiltration tools).**

  There are several types of malicious payloads used by attackers, each tailored to achieve different objectives. Backdoors are designed to give attackers continuous access to the compromised system, often evading detection to maintain long-term control. Keyloggers record the keystrokes of users to capture sensitive information, such as passwords or credit card numbers, which can be used for identity theft or unauthorized access. Data exfiltration tools are specialized payloads used to quietly extract valuable data from the victim's system, typically sensitive corporate data, intellectual property, or personal information. Each type of payload can be customized to target specific systems or bypass particular security measures, making them versatile tools in a hacker's arsenal.

1.5.4 Payloads in networking

- **Payload in data packets (OSI model).**

  In the OSI (Open Systems Interconnection) model, the payload is the actual data that needs to be transmitted across the network, residing in the "Data" section at different layers. It's encapsulated by headers and footers as it moves down the OSI layers—from the application layer, where the data originates, to the physical layer, where it's transmitted as electrical signals or light. The payload is essentially the core information that needs to reach its destination, such as a message, file, or streaming data. As the packet moves through each layer, additional information, such as addressing and error-checking codes, is appended to ensure successful delivery, but the payload remains unchanged. Its size and format can vary depending on the protocol in use, with specific layers responsible for fragmenting and reassembling larger payloads to fit transmission limits.

- **Distinction between payload and headers in networking**
  The distinction between payload and headers in networking lies in their function and structure. The payload is the user data being transported, while the headers contain control information that facilitates the accurate delivery of this payload. Headers include metadata like the source and destination addresses, sequence numbers, and error-checking data. For example, in an IP packet, the header contains the IP addresses of the sender and receiver, ensuring the packet is routed correctly, while the payload is the actual message or data being sent. Headers enable the network to handle issues like congestion control, routing, and reliability, while the payload is the content the end-users care about. In essence, headers ensure that the payload reaches its intended destination correctly, while the payload is the reason the transmission occurs in the first place.

1.5.5 Payload Capabilities in Software

- **Payload capabilities in penetration testing frameworks (e.g., Metasploit, Cobalt Strike).**

  Penetration testing frameworks like Metasploit and Cobalt Strike offer extensive payload capabilities that allow security professionals to simulate real-world attacks. In Metasploit, payloads are classified into different types, such as singles, stagers, and stages, providing flexibility in how attacks are crafted. For example, staged payloads allow attackers to send a small initial payload that later downloads and executes a larger, more complex payload, making it harder to detect. Cobalt Strike, on the other hand, excels in post-exploitation payloads with its Beacon payload, which is used to maintain long-term control over a compromised machine through features like file transfer, keylogging, and privilege escalation. Both platforms offer customizable payloads that adapt to various attack scenarios, enabling red teams to assess how vulnerable an organization is to different forms of exploitation.

- **Modular payload systems in software.**

Modular payload systems in penetration testing frameworks are designed to be highly flexible and reusable. Rather than using a monolithic approach, where a single payload carries out all attack functions, modular payloads allow different components—such as exploits, listeners, and post-exploitation modules—to be combined dynamically. In Metasploit, for example, a tester can select an exploit targeting a specific vulnerability and pair it with a payload suited to the post-compromise goal, such as executing a reverse shell. This modularity ensures that the same exploit can be used with different payloads depending on the context of the test. This design reduces redundancy, allowing security experts to tailor attacks more precisely and respond quickly to changes in security environments, making it a powerful feature in offensive security toolkits.

- **Payload customization for different targets (operating system, architecture).**

  Customization of payloads for different targets is crucial in penetration testing, as each system has its own architecture, operating system, and security posture. Attackers or penetration testers must consider these factors when choosing or creating a payload. For example, a payload designed for a Windows system won't work on Linux or macOS due to differences in system calls and architecture. Similarly, payloads targeting 32-bit architectures will fail on 64-bit machines unless specifically designed to handle both. Penetration testing frameworks like Metasploit allow testers to customize payloads to match these variables, including adapting the payload to evade detection by specific antivirus software or network defenses. This versatility is essential for ensuring that the attack vector remains effective regardless of the target environment, making payload customization a key aspect of successful penetration testing.

1.5.6 Payload Capabilities in Collecting Data

Different types of malwares are designed to collect sensitive data from a user's computer and provide it to threat actors. This category includes malware like spyware and adware.

Spyware

Spyware is tracking software installed without the user's knowledge or consent. It typically monitors user activity in secret, gathering information using the computer's existing resources and programs. This data is then collected and distributed without the user's approval, often involving personal or sensitive information. One particularly malicious type of spyware is a keylogger. Keyloggers quietly record every keystroke a user types on their keyboard, enabling attackers to sift through the captured data for valuable details such as passwords, credit card numbers, or personal information. And Software Keyloggers shown in Figure 1-8

Fig [1-8] Shows Software Keylogger.

Early keyloggers were hardware devices that were inserted between the computer's keyboard connection and the USB port, as shown in Figure 1-9. These devices looked like ordinary plugs and were often connected to the USB port at the back of the computer, making them difficult to detect. Since they are physical devices, they are not scanned by antimalware software, allowing them to operate undetected. However, because attackers must return to physically retrieve the hardware keylogger to access the captured information, hardware keyloggers are not commonly used today.

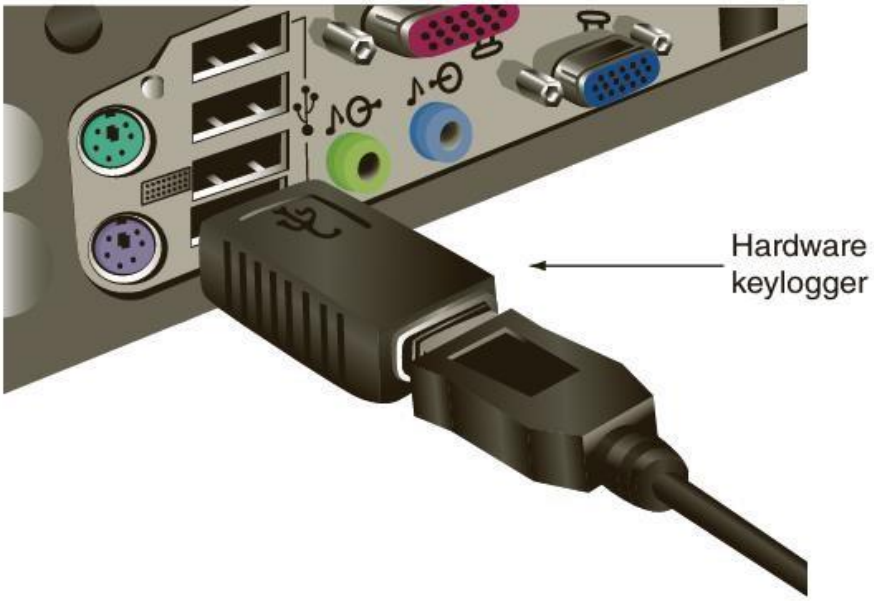


Fig [1-9] Shows hardware Keylogger.

Note

Not all spyware is necessarily malicious. For example, spyware monitoring tools can help parents keep track of the online activities of their children

Adware

Adware delivers advertising content in ways that are unexpected and unwanted by users. Once installed, adware typically displays banners, pop-up ads, or opens new browser windows at random intervals. Users generally dislike adware for several reasons:

- It can display inappropriate content, such as gambling or pornography.
- Frequent pop-up ads disrupt productivity.
- These ads can slow down a computer or even cause crashes, potentially leading to data loss.
- The sheer volume of ads is often annoying.

As online advertising has increased, users have responded by installing ad-blocking software in their browsers. Ad blocking grew by 41% worldwide within a year, reaching 198 million users, while U.S. ad blocking rose by 48%, impacting 45 million users. It's estimated that ad blocking costs website publishers around $22 billion annually. To address this, marketers have started implementing paywalls (requiring users to pay to access content without ads), friendly reminders about the purpose of ads (highlighting that ads fund free content), and warnings that content may be blocked if ad blockers are detected.

1.5.7 Payload Capabilities in Deleting Data

Logic Bomb

While spyware and adware are designed to collect data, other types of malwares are built to do the opposite—delete it. This can involve erasing important user files, such as documents or photos, or removing crucial operating system files, rendering the computer unusable.

One common type of malware used for data deletion is a logic bomb. A logic bomb is malicious code embedded within a legitimate program that remains inactive until triggered by a specific event. Once activated, the logic bomb can delete data or carry out other harmful actions. For instance, a government employee in Maryland attempted to destroy data on over 4,000 servers by planting a logic bomb set to detonate 90 days after his termination.

A notable recent attack using a logic bomb targeted three banks and two media companies in South Korea. The malware contained four files, including AgentBase.exe, which initiated the attack. Embedded within the file was a hexadecimal string (4DAD4678) representing the date and time of the attack:

March 20, 2013, at 2:00 PM. Once the system clock hit 2:01 PM, the logic bomb activated, overwriting the hard drive and master boot record on Microsoft Windows computers, causing them to reboot and become unusable. The malware also had a module to delete data from remote Linux machines. Detecting logic bombs before they are triggered is challenging. They are often hidden within large programs containing tens of thousands of lines of code, and a trusted insider can easily add a few lines of malicious code without raising suspicion. Additionally, these programs are not routinely scanned for malicious content.

1.5.8 Payload Capabilities in Modifying System Security

Backdoor

Some types of malware aim to alter a system's security settings to enable more dangerous attacks. One such type of malware is known as a backdoor. A backdoor provides unauthorized access to a computer, program, or service, bypassing normal security protections. Once installed, it allows attackers to return later and evade security measures.

Note

Developers often create legitimate backdoors to access a program or device regularly without being interrupted by repeated password prompts or security checks. These backdoors are intended to be removed once the application is complete. However, in some cases, backdoors have been left in place, allowing attackers to exploit them and bypass security.

1.5.9 Payload Capabilities in Launching Attacks

Botnet

A popular malware payload today allows an infected computer to be remotely controlled by an attacker to carry out attacks. These infected machines, known as bots or zombies, are often part of a larger network called a botnet, which is controlled by a bot herder. Botnets can consist of hundreds, thousands, or even millions of compromised computers.

Bot computers receive instructions from the bot herder through a command-and-control (C&C) structure. This communication can occur in several ways, including:

- Bots may automatically log in to a website run by the bot herder, where commands are posted for them to interpret and execute.
- Bots can log in to third-party websites, allowing the bot herder to remain unaffiliated with the site.
- Commands can be delivered through blogs, Twitter posts, or Facebook notes, using specially coded instructions.
- Increasingly, bot herders use a "dead drop" C&C method, where they create a Google Gmail account and leave commands in a draft email that is never sent. Bots log in to the account, read the draft, and receive the instructions. Since the email is never sent, there is no record of it, and Gmail's encryption ensures the transmission remains private.

The scale of botnets is staggering. According to the FBI’s cyber division, 18 computers are infected and added to botnets every second, resulting in hundreds of millions of compromised machines each year. Some botnets control 3 to 4 million bots, while another botnet used for email spam was responsible for sending up to 60 billion emails daily.

1.6 Malware's Dynamic Analysis vs. Static Analysis

1.6.1 What's the Difference?

Dynamic analysis involves analyzing a program while it is running, allowing for the examination of its behavior in real-time. This method is useful for detecting runtime errors and performance issues. On the other hand, static analysis involves examining the code without executing it, focusing on identifying potential bugs and vulnerabilities before the program is run. While dynamic analysis provides more accurate results, static analysis is more efficient and can be performed earlier in the development process. Both methods have their strengths and weaknesses, and a combination of both is often used to ensure comprehensive testing of software.

Attribute	Dynamic Analysis	Static Analysis
Timing	Occurs during runtime	Occurs before runtime
Code execution	Code is executed	Code is not executed
Performance impact	May slow down the system	No performance impact
Identifying bugs	Can find runtime bugs	Can find syntax errors and potential bugs

Automation	Can be automated	Cannot be automated

1.6.2 Further Details

1.6.2.1 Introduction

Dynamic analysis and static analysis are two common techniques used in software testing to identify defects and vulnerabilities in software applications. While both methods aim to improve the quality and security of software, they differ in their approach and the types of issues they can uncover. In this article, we will compare the attributes of dynamic analysis and static analysis to help you understand their strengths and weaknesses.

1.6.2.2 Dynamic Analysis

Dynamic analysis, also known as black-box testing, involves executing the software application and observing its behavior in real-time. This technique simulates the actual runtime environment of the software and can uncover issues related to performance, memory leaks, and security vulnerabilities that may only manifest during execution. Dynamic analysis tools typically include profilers, debuggers, and fuzzers that help testers identify and diagnose problems in the software.

- Executes the software application to observe its behavior.
- Simulates the actual runtime environment.

- Uncover issues related to performance, memory leaks, and securityvulnerabilities.
- Includes profilers, debuggers, and fuzzers.

1.6.2.2 Static Analysis

Static analysis, on the other hand, is a white-box testing technique that involves examining the source code or binary of the software without executing it. This method focuses on identifying issues such as coding errors, security vulnerabilities, and compliance violations by analyzing the code structure, syntax, and dependencies. Static analysis tools use algorithms and rules to scan the codebase and generate reports on potential issues that may exist in the software.

- Examines the source code or binary without executing it .
- Focuses on identifying coding errors, security vulnerabilities, and compliance violations.
- Analyzes the code structure, syntax, and dependencies.
- Uses algorithms and rules to scan the codebase.

1.6.2.3 Comparison

Dynamic analysis and static analysis have their own strengths and weaknesses when it comes to software testing. Dynamic analysis is effective at uncovering runtime issues and performance bottlenecks that may not be apparent during static analysis. It can also help identify security vulnerabilities that only manifest when the software is running. However, dynamic analysis can be time-consuming and may not provide a comprehensive view of all possible issues in the software. On the other hand, static analysis is great for identifying coding errors and security vulnerabilities early in the development process. It can help developers catch issues before they become critical problems and improve the

overall quality of the codebase. Static analysis is also faster than dynamic analysis since it does not require the software to be executed. However, static analysis may produce false positives and may not catch all runtime issues that dynamic analysis can uncover.

1.6.2.4 Conclusion

In conclusion, both dynamic analysis and static analysis are valuable techniques in software testing that offer unique benefits to developers and testers. Dynamic analysis is great for uncovering runtime issues and security vulnerabilities that may only manifest during execution, while static analysis is effective at identifying coding errors and security vulnerabilities early in the development process. By combining both techniques in a comprehensive testing strategy, developers can ensure that their software is of the highest quality and security standards.

1.7 Malware's Real-life examples

1.7.1 The Target Data Breach (2013)

One of the most infamous real-life examples of a malware and social engineering attack occurred during the Target data breach in 2013. Cybercriminals used a phishing attack to trick a third-party vendor, gaining access to Target's network. Once inside, the attackers deployed malware to capture credit card information from over 40 million customers at point-of-sale terminals. This attack combined social engineering to manipulate an unsuspecting user into granting access and malware to stealthily extract sensitive financial data from Target's systems, all while remaining undetected for several weeks.

Video Explanation: (196) Target 2013 Data Breach Explained - YouTube

1.7.2 The Target Data Breach (2013)

The WannaCry ransomware attack in 2017 was a global cyberattack that affected thousands of organizations, including hospitals, businesses, and government agencies. WannaCry spread through a vulnerability in Microsoft's Windows operating system, encrypting files on infected computers and demanding ransom payments in Bitcoin to unlock them. In this case, the malware didn't rely on sophisticated social engineering but exploited a known system vulnerability. However, many victims were tricked into opening malicious email attachments that initiated the attack. The rapid spread of the malware across networks demonstrated how damaging a malware attack can be when paired with weak cybersecurity defenses and social engineering to trigger initial infection.

Video Explanation: WANNACRY: The World's Largest Ransomware Attack (Documentary) - YouTube

1.7.3 The Twitter Bitcoin Scam (2020)

In 2020, several high-profile Twitter accounts, including those of Barack Obama, Elon Musk, and Bill Gates, were compromised in a social engineering attack that led to a widespread Bitcoin scam. Hackers used social engineering tactics to trick Twitter employees into giving access to internal tools. Once the attackers gained control, they tweeted messages from the compromised accounts, promising to double any Bitcoin sent to a specific address. While this attack didn't involve malware, it was a stark example of how social engineering can be leveraged to exploit trusted platforms and cause financial damage on a massive scale, affecting both individuals and the companies involved.

Video Explanation: The Teenager Who Hacked Twitter And Stole Millions In Bitcoin (youtube.com)

1.7.4 Stuxnet (2010)

Stuxnet is a highly targeted worm designed to infiltrate industrial control systems, specifically those in Iranian nuclear facilities. It spread via infected USB drives and targeted PLCs (programmable logic controllers), causing physical damage by altering the speed of centrifuges in nuclear plants. This attack, believed to be state-sponsored by the US and Israel, significantly set back Iran's nuclear program and marked the first publicly known instance of malware engineered to harm physical infrastructure. Defense strategies include isolating critical systems from the internet through air-gapping and raising cybersecurity awareness in industrial environments.

Video Explanation: Stuxnet: The Cyber Weapon That Destroyed Iran's Nuclear Program (youtube.com)

1.7.5 Emotet (2014)

Emotet, first identified in 2014 as a banking Trojan, initially targeted financial institutions but later evolved into a botnet that infected a wide range of industries. It spread primarily through phishing emails containing malicious attachments or links, harvesting banking credentials and facilitating the distribution of other malware, such as ransomware. Emotet's widespread impact included affecting businesses and organizations globally, with its resilience making it difficult to detect and remove. Over time, it became a robust botnet infrastructure responsible for significant financial losses. Effective defenses include strong email filtering, robust anti-malware solutions, and thorough employee education on phishing threats.

Video Explanation: What is Emotet? (youtube.com)

1.7.6 Zeus (2007)

Zeus, also known as Zbot, is a notorious Trojan horse first identified in 2007, primarily designed to target Windows systems for the purpose of stealing banking information. Zeus operated by infiltrating a victim's computer and silently monitoring their activities, particularly focusing on capturing sensitive financial data through keylogging and form-grabbing techniques. Keylogging allowed Zeus to record every keystroke made by the user, enabling it to capture login credentials, passwords, and other personal information as they were typed. Form-grabbing, on the other hand, intercepted data submitted through web forms, such as those used on banking websites, before it could be encrypted and securely transmitted. Zeus was highly effective due to its stealthy nature, often spreading through phishing emails, malicious downloads, and compromised websites. It infected millions of computers worldwide, creating a vast botnet used

to siphon off billions of dollars from bank accounts. The malware was so successful that it gave rise to various offshoots and derivatives, making Zeus one of the most impactful pieces of malware in cybercrime history. Despite numerous takedowns and arrests related to Zeus, its legacy continues to influence modern banking Trojans.

Video Explanation: The First Virus to Infiltrate Banks and Steal Millions, Zeus.exe (youtube.com)

1.7.7 AgentTesla (2014)

AgentTesla, first discovered in 2014, is a powerful and widely used spyware and keylogger designed to steal sensitive information from compromised systems. This malware primarily targets Windows environments and is often distributed through phishing campaigns, which entice users to download malicious attachments or click on links that execute the malware. Once installed, AgentTesla operates stealthily in the background, capturing keystrokes, taking screenshots, and logging clipboard data. It is particularly effective at stealing login credentials, including those for email accounts, VPNs, and other critical applications. AgentTesla is also capable of extracting information from web browsers, email clients, and FTP servers. The malware communicates the stolen data back to the attacker via HTTP POST requests or through SMTP, making it difficult to detect. Its modular design allows it to be customized for specific attacks, and it has been sold as a "malware-as-a-service" on underground forums, making it accessible to a wide range of cybercriminals. AgentTesla has been involved in numerous high-profile attacks, often targeting businesses to gain access to corporate networks and steal valuable data, leading to significant financial and reputational damage for the victims. Despite efforts to curb its spread, AgentTesla remains a persistent threat due to its adaptability and the ongoing demand for its capabilities among cybercriminals.

Video Explanation: Agent Tesla (youtube.com)

1.7.8 NotPetya (2017)

NotPetya, discovered in June 2017, is a form of wiper malware that initially masqueraded as ransomware, targeting Windows systems and causing widespread devastation across the globe. Originating in Ukraine, NotPetya exploited a vulnerability in Microsoft Windows known as EternalBlue, which allowed it to propagate rapidly within networks. Unlike traditional ransomware, NotPetya had a destructive purpose; it aimed to wipe out data rather than encrypt it for ransom, rendering infected systems inoperable. Once executed, NotPetya encrypted files on the infected machine, but it also corruptly overwrote the Master File Table (MFT), making data recovery nearly impossible without extensive backups.
The malware spread quickly through various vectors, including compromised software updates and the use of legitimate administrative tools like PsExec. NotPetya's impact was particularly severe, affecting major corporations, government entities, and critical infrastructure worldwide, leading to billions of dollars in damages. Notable victims included Maersk, Merck, and FedEx, all of which faced significant operational disruptions due to the malware's rampage. The sophistication of NotPetya highlighted the growing trend of cyber warfare, with many cybersecurity experts attributing its development to state-sponsored actors, particularly in the context of geopolitical tensions. The aftermath of NotPetya underscored the importance of robust cybersecurity measures, regular backups, and the need for organizations to prepare for advanced persistent threats that could lead to catastrophic data loss.

Video Explanation : NotPetya Attack (youtube.com)

Note

The Master File Table (MFT) is a critical component of the NTFS (New Technology File System) used by Windows operating systems. It acts as a database that contains information about every file and directory on the NTFS volume, including file names, sizes, locations, attributes, and data runs. Each file and directory have a corresponding entry in the MFT, allowing the operating system to quickly access and manage files

Chapter 2: SIEM Configuration and Monitoring

2.1 Outline

1. What is SIEM?
2. Mention Components of SIEM.
3. How SIEM Components Work?
4. Mention list of open-source SIEM tools.
5. Mention how the open-source SIEM tools work.
6. Mention What Is Wazuh open Source.
7. How Wazuh Is Work?
8. How To Build Wazuh SIEM Open Source?
9.What Is Wazuh-Agent?
10.What is putty?

2.2 What's SIEM?

SIEM (Security Information and Event Management) is a comprehensive security solution that combines two essential aspects of cybersecurity: Security Information Management (SIM) and Security Event Management (SEM). The primary function of a SIEM is to provide organizations with real-time visibility into their security environment by collecting, analyzing, and correlating security data from multiple sources.

Key Components of SIEM: -

2.2.1 Log Collection:

SIEM systems gather log data from various sources like servers, applications, network devices, firewalls, intrusion detection systems (IDS), and more. These logs contain records of events, errors, and user activity.

2.2.2 Event Correlation:

The SIEM platform correlates the data by identifying patterns or relationships between different events and logs. This helps in detecting security threats that may not be evident in individual logs but become clear when data from various sources is combined.

2.2.3 Real-Time Monitoring:

SIEM continuously monitors network activity, logs, and security events in real time. It identifies abnormal behavior and potential security threats like unauthorized access, malware activity, or policy violations.

2.2.4 Incident Detection and Response:

Based on the analysis of logs and event data, SIEM detects security incidents and can automatically trigger alerts to security teams. In some cases, SIEM can automate the response by taking predefined actions (e.g., blocking an IP address or isolating a compromised system).

2.2.5 Alerting and Reporting:

SIEM systems generate alerts based on predefined rules and thresholds, enabling security teams to respond to potential threats quickly. The system also produces detailed reports for compliance, audits, and forensics.

2.2.6 Threat Intelligence Integration:

Many SIEMs integrate threat intelligence feeds to stay updated with the latest cyber threats, vulnerabilities, and attack techniques. This allows the system to detect emerging threats more efficiently.

2.2.7 Compliance and Audit Support:

SIEM helps organizations comply with various regulatory requirements, such as GDPR, HIPAA, and PCI-DSS, by logging and reporting security events, ensuring that they meet audit requirements and maintain security best practices.

2.3 Benefits Of SIEM

- Centralized Security Management: SIEM systems provide a single platform for monitoring and managing security data across the entire IT infrastructure.
- Improved Incident Response: SIEM solutions enable faster detection of security threats and reduce the response time by automating alerting and response mechanisms.
- Enhanced Threat Detection: By correlating data from multiple sources, SIEM systems can identify advanced and complex attacks that may go unnoticed otherwise.
- Compliance: SIEM helps organizations meet regulatory compliance by maintaining security logs and generating reports for auditors.

2.3.1 Use Case Example:

Imagine a company that has hundreds of servers, firewalls, and endpoints generating logs. A SIEM system collects logs from all these devices and correlates them to detect suspicious activities, such as multiple failed logins attempt across different systems, which could indicate a brute-force attack. It then generates an alert for the security team to investigate, preventing potential breaches.

2.3.2 Conclusion:

SIEM is a vital tool for modern cybersecurity management, offering visibility, real-time detection, incident response, and compliance support. By centralizing security data and applying intelligent analysis, it helps organizations detect and respond to threats before they cause significant damage.

2.4 How The SIEM Works?

2.4.1 Data Collection:

- SIEM gathers logs and event data from various sources, such as:

o Network devices (routers, switches).

o Servers (Windows, Linux, etc.).

o Security tools (firewalls, IDS/IPS, antivirus).

o Applications (web applications, databases).

o Cloud environments.

- This data includes login attempts, traffic patterns, system errors, access records, and more.
- Data is collected via log agents installed on these devices or through syslog protocols.

2.4.2 Data Normalization:

- Collected data comes in different formats, so the SIEM system normalizes it to ensure consistency.
- Normalization involves converting different log formats into a standard format that can be analyzed. For example, a login attempt on a Windows server might look different from one on a Linux server, but normalization makes them comparable.

2.4.3 Data Aggregation:

- The SIEM system aggregates similar logs or events into a unified view.
- Instead of processing individual log events separately, the system combines related events (e.g., repeated failed login attempts) to make analysis more efficient and reduce unnecessary data.

2.4.4 Data Correlation:

- The correlation engine is the core component of SIEM. It examines multiple events across different systems and correlates them to detect security incidents.

For example, a single failed login attempt might not be a problem, but multiple failed login attempts followed by a successful login could indicate a brute-force attack.

- Correlation rules are defined based on security policies, threat intelligence, or known attack patterns (like recognizing attempts to exploit vulnerabilities).

2.4.5 Threat Detection:

- SIEM uses correlation rules and threat intelligence feeds to detect potential security threats in real-time.
- If the system detects an anomaly, such as abnormal login times, unusual network traffic, or multiple authentication failures, it flags this as a potential security incident.

2.4.6 Alerting:

- When a security incident or anomaly is detected, the SIEM system triggers an alert.
- Alerts are sent to security teams through emails, dashboards, or even integrated ticketing systems.
- SIEM can categorize alerts based on severity (e.g., low, medium, high), allowing teams to prioritize which incidents to respond to first.

2.4.7 Incident Response:

- Many SIEM systems have automated response capabilities. Based on predefined rules, the system can take actions such as:
  - o Blocking IP addresses.
  - o Isolating infected machines from the network.
  - o Stopping specific processes or services.

- Incident response teams can also manually investigate the alerts to determine if further action is needed.

2.4.8 Log Storage and Retention:

- SIEM systems store logs for future analysis, audits, and compliance reporting.
- Logs are stored for a defined period (e.g., months or years), based on organizational policies or compliance requirements (such as PCI-DSS, HIPAA).
- Having historical data helps with investigations and forensics to understand how a breach or attack occurred.

2.4.9 Reporting and Dashboards:

- SIEM systems provide customizable reports and dashboards for security monitoring.
- These reports are used for:
  - o Compliance auditing (meeting regulatory requirements).
  - o Detecting trends over time.
  - o Security posture reviews.

2.4.10 Threat Intelligence Integration:

- SIEM systems often integrate with threat intelligence feeds to stay updated on the latest vulnerabilities, malware, and attack vectors.
- These feeds help detect new types of threats, allowing the SIEM to identify and respond to emerging risks in real-time.

2.4.11 Examples: How SIEM Works in a Cybersecurity Incident:

1. Data Collection: Logs from a firewall show that there's been an unusually high number of inbound traffic requests to a web server.

2. Data Normalization and Correlation: The SIEM system correlates these firewall logs with logs from the web server, which show multiple failed login attempts followed by a successful one from the same IP address.

3. Alerting: The SIEM generates an alert, flagging the sequence of events as a potential brute-force attack followed by unauthorized access.

4. Incident Response: Based on predefined rules, the SIEM system could automatically block the suspicious IP address or alert the incident response team for immediate investigation.

5. Reporting: Security analysts can review the logs and correlation data via the SIEM dashboard, generate a report for further analysis, and take appropriate actions to prevent further damage.

2.5 Open Source SIEMs:

There are several open-source SIEM (Security Information and Event Management) tools available that offer robust features for log management, threat detection, and security analytics. Here are some popular ones:

2.5.1 ELK Stack (Elastic Stack)

- Components: Elasticsearch, Logstash, and Kibana.
- Features:

o Real-time log collection and analysis.
o Customizable dashboards and visualization.
o Supports integrations with Beats and other log shippers.

- Use Case:

Widely used for log management and as a SIEM tool by configuring alerting (e.g., through Elastic's free tier or integrations).

- Link: Elastic Stack: (ELK) Elasticsearch, Kibana & Logstash | Elastic

2.5.2 Wazuh

- Features:

o Intrusion detection, log analysis, and incident response.
o File integrity monitoring (FIM) and vulnerability detection.
o Real-time alerting and compliance monitoring.

- Use Case:

Open-source platform built on top of Elastic Stack; acts as a SIEM solution.

- Link: Wazuh - Open Source XDR. Open Source SIEM.

2.5.3 Graylog

- Features:
o Centralized log collection and management.
o Customizable alerts and dashboards.
o Integrates with Elasticsearch for data storage.

- Use Case: Known for its performance in managing large-scale log data.

- Link: SIEM, Log Management & API Protection (graylog.org)

2.4.4 Security Onion

- Features:
o Intrusion detection, network monitoring, and log management.
o Uses a combination of tools like Zeek (formerly Bro), Suricata, and Elasticsearch.
o Incident response and threat hunting capabilities.

- Use Case: Designed for network security monitoring and intrusion detection but can also serve as a SIEM.

- Link: Security Onion Solutions

2.4.5 The Hive

• Features:
o Incident response platform with case management and collaboration.
o Can integrate with Cortex for threat intelligence analysis and automated response.
o Threat indicators and IoC (Indicators of Compromise) management.

• Use Case: Incident response platform that works well with other SIEM tools.

• Link: StrangeBee

2.4.6 Apache Metron

• Features:
o Real-time security data analytics.
o Pluggable for custom threat intelligence feeds.
o Built on top of Apache Hadoop and supports large-scale processing.

• Use Case: Suitable for organizations that need big data security analytics.

• Link: Apache Metron Big Data Security

2.5.7 SIEMonster

• Features:
o End-to-end open-source SIEM solution built on a range of open-source tools. to Scalable, from small business to large enterprises.
o Incorporates Elastic Stack, Wazuh, and others for monitoring and detection.

• Use Case: Provides a complete SIEM solution with multi-tenant and dashboard features.

• Link: Home - SIEMonster

2.5.8 AlienVault OSSIM

• Features:
o Combines SIEM and threat detection with vulnerability assessment.
o Log management, asset discovery, and network monitoring. o Real-time event correlation and security monitoring.

• Use Case: A solid open-source platform for small to mid-sized organizations.

• Link: Enhance Security with OSSIM | LevelBlue (att.com)

2.5.9 Prelude SIEM

• Features:
o Open-source hybrid SIEM.
o Real-time data collection, correlation, and normalization.
o Focuses on interoperability with other security tools.

• Use Case: Designed to be an extensible and interoperable SIEM solution.

• Link: Overview - PRELUDE SIEM (prelude-ids.org)

2.5.10 Snort (with additional tools)

- Features:
o Primarily a network intrusion detection system (NIDS).
o Can be integrated with other tools like ELK Stack or Wazuh for log management and threat detection.

- Use Case: Use alongside SIEM platforms for network-based threat detection.

- Link: Snort - Network Intrusion Detection & Prevention System

---

These tools provide a wide range of options for different needs, from large-scale log management and monitoring to complete SIEM solutions with integrated threat detection and response capabilities. Depending on your organization's size, infrastructure, and specific needs, you can choose the one that best fits your environment and different open-source SIEMs shows in figure 2-1.

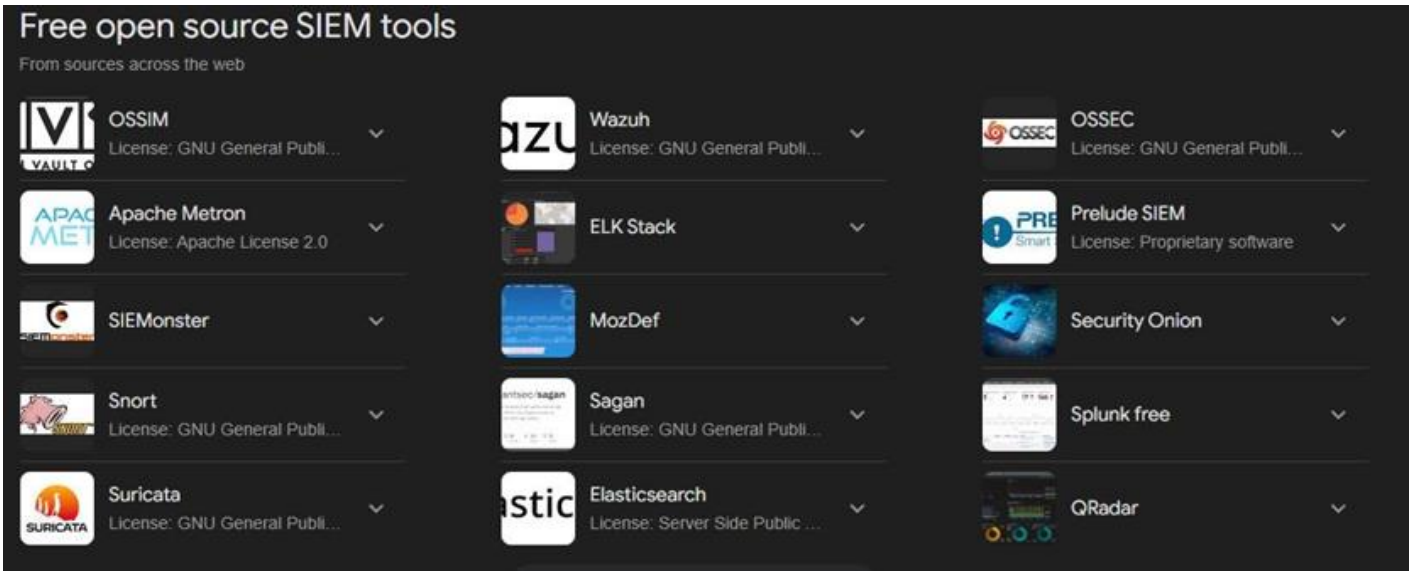


Fig [2-1] Shows different popular open-source SIEMs that SOC analysts and engineers used.

2.6 How Open-Source SIEM Tools Work?

- ***Log Collection: Most tools (e.g., ELK, Graylog, Wazuh) collect logs from various sources such as system logs, network devices, and applications.***
- ***Log Analysis: Logs are processed, normalized, and correlated to detect anomalies or suspicious patterns.***
- ***Threat Detection: Signature-based tools like Snort or behavior-based tools like Zeek help identify potential threats in the data.***
- ***Visualization: Dashboards and visualizations (e.g., Kibana, Metron Dashboard) provide real-time monitoring and analytics.***
- ***Alerting: Customizable alerts notify you of security incidents or anomalies in real- time.***
- ***Incident Response: Some tools like TheHive focus on managing security incidents, while others (e.g., Wazuh, Security Onion) integrate incident response capabilities into their core functions.***

These SIEM tools provide a combination of log management, threat detection, and incident response through efficient collection, analysis, and real-time monitoring of security data.

2.7 What's Wazuh?

Wazuh is an open-source security monitoring and data analysis system, designed to collect and analyze data from various systems to provide a comprehensive view of security threats and compliance. It is considered the successor to OSSEC, having been developed and enhanced to include advanced capabilities in log analysis, threat detection, and incident response

.

Wazuh Features:

2.7.1 Log Management

o Collects logs from multiple devices such as servers, applications, and network devices.
o Allows for the analysis of these logs to detect unusual activities or potential threats.

2.7.2 Threat Detection

o Relies on predefined security rules to detect intrusion attempts, vulnerabilities, or malicious activities.
o Provides capabilities to detect malware, breaches, and unauthorized access attempts.

2.7.3 Compliance Monitoring

o Assists in compliance with security and privacy standards such as GDPR, HIPAA, and PCI DSS, by monitoring systems and generating the required reports.

2.7.4 SIEM Integration

o Integrates with SIEM tools like Elastic Stack (ELK) and Splunk to provide deeper log and threat analysis.

2.7.5 Multi-host Monitoring

o Can monitor hundreds or thousands of devices and systems from a central location.
o Operates on a client/server model, where agents collect data and send it to the central server for analysis.

2.7.6 Alerting

o Can send real-time alerts when suspicious activities or threats are detected.
o Supports integration with other alerting systems such as Slack, Email, and Webhook.

2.8 How Wazuh Works?

2.8.1 Agents

o Agents are installed on various systems (servers, desktops, applications) to collect logs and data.

2.8.2 Server

o The central server collects data from agents, analyzes it using predefined security rules, and generates reports and alerts based on the findings.

2.8.3 Dashboard

o Uses a Kibana-based interface (visual tool) to display information and analysis in an easy-to-understand and visual format.

Wazuh Use Cases:

- ***Network Security Monitoring: Monitoring networks and detecting security threats.***
- ***Incident Response: Providing alerts and quick responses to security incidents.***
- ***Log Analysis: Comprehensive log analysis for security improvement and compliance reporting.***

Wazuh is an effective and powerful tool for monitoring systems and security in organizations, both large and small, providing a comprehensive and analytical view of security events and helping to enhance overall security, and Figure 2-2 Show Wazuh's Dashboard

---

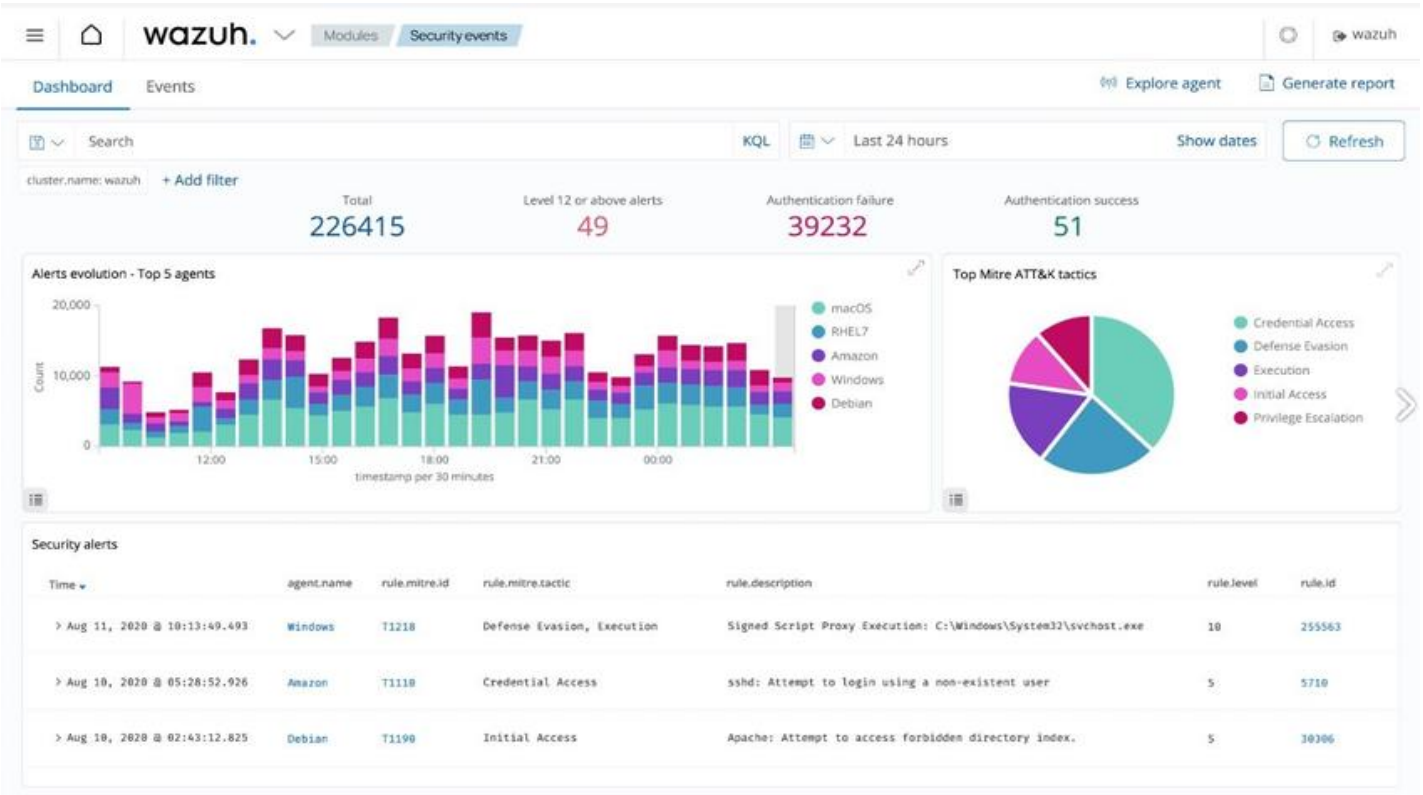


Fig [2-2] Shows Wazuh's Dashboard, Describes the alerts specially monitored security alerts.

2.9 How to Build Wazuh?

Building a Wazuh SIEM involves setting up the Wazuh server, agents, and an interface for data visualization. Here's a step-by-step guide on how to build Wazuh SIEM:

2.9.1 Prerequisites:

1- A Linux server (Ubuntu, Debian, CentOS, etc.).

2- Minimum hardware requirements:
o 2 CPU cores.
o 4 GB of RAM.
o 20 GB of storage (may vary depending on log volume).

3- Root or sudo access.

2.9.2 Install Wazuh Server:

The Wazuh server is responsible for collecting data from agents, analyzing it, and generating alerts. You can install it on a Linux server. Follow these steps:

Download the Wazuh OVA File

1- ***Visit the Wazuh website and download the latest Wazuh OVA file. The OVA contains preconfigured components like Wazuh Manager, Wazuh Indexer (formerly Elasticsearch), and the Wazuh Dashboard (formerly Kibana).***

2- ***Install VMware Workstation or VMware Player If you don't have VMware Workstation or VMware Player installed, download and install the appropriate version for your OS:***

- ***VMware Workstation Player (Free for personal use) or VMware Workstation Pro.***

3. Import Wazuh OVA into VMware

- ***Open VMware Workstation or Player:***
  - ***Open VMware, go to File > Open.***

- ***Select the OVA file:***
  - ***Browse and select the Wazuh OVA file you downloaded and click Open.***

4. Review Virtual Machine Settings:
- ***Review the OVA's configuration (RAM, CPU, disk size, etc.). You can adjust the settings (e.g., increasing RAM or CPUs) based on your system's resources and requirements.***

5. Import the Virtual Machine:
o Click Import to begin importing the OVA. This process may take a few minutes depending on your system.

6. Start the Wazuh Virtual Machine
- Once the import is complete, you will see the Wazuh VM in the list.
- Select the Wazuh VM and click Start (or play virtual machine).
- The VM will boot, and Wazuh services (Wazuh Manager, Wazuh Indexer, and Wazuh Dashboard) will automatically start. It may take a couple of minutes for all services to be fully operational.

<u>Three Main Commands to Run: -</u>

- ***sudo systemctl start wazuh-manager***
- ***sudo systemctl start wazuh-dashboard***
- ***sudo systemctl start wazuh-indexer***

And these Commands Shows in Figure 2-3.

```
[wazuh-user@wazuh-server ~]$
[wazuh-user@wazuh-server ~]$ sudo systemctl start wazuh-manager
[wazuh-user@wazuh-server ~]$ sudo systemctl start wazuh-dashboard
[wazuh-user@wazuh-server ~]$ sudo systemctl start wazuh-indexer
[wazuh-user@wazuh-server ~]$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP gr
oup default qlen 1000
    link/ether 00:0c:29:e8:b1:da brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.231/24 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 85956sec preferred_lft 85956sec
    inet6 fe80::20c:29ff:fee8:b1da/64 scope link
       valid_lft forever preferred_lft forever
[wazuh-user@wazuh-server ~]$
[wazuh-user@wazuh-server ~]$
[wazuh-user@wazuh-server ~]$ sudo systemctl start wazuh-manager
[wazuh-user@wazuh-server ~]$ sudo systemctl start wazuh-dashboard
[wazuh-user@wazuh-server ~]$ sudo systemctl start wazuh-indexer
[wazuh-user@wazuh-server ~]$
```

Fig [2-3] Shows Three Main Commands and Components of Wazuh dashboard.

Once the Wazuh VM is running, it will display its IP address (usually via DHCP).
To access the Wazuh Dashboard:
1. Open a web browser.
2. Navigate to:

```
https://<WAZUH_VM_IP>:5601
```

Fig [2-4] Shows the Recommended IP and Port to connect your host to wazuh server

Copy code
https://<WAZUH_VM_IP>:5601
Replace <WAZUH_VM_IP> with the actual IP of the Wazuh VM.

3. Bypass any SSL warnings (self-signed certificates).

4. Login using default credentials:
o Username: admin
o Password: admin

Fig [2-5] Shows Wazuh Cardinalates Login to appear dashboard

2.10 Key Features of Wazuh Dashboard:

2.10.1 Real-time Monitoring

o Provides real-time monitoring of your systems and security events. It allows you to view the status of agents, analyze logs, and detect anomalies as they occur.

2.10.2 Security Events Visualization

o The dashboard allows users to create customizable visualizations and dashboards to track events, visualize trends, and monitor various security metrics such as threats, vulnerabilities, and compliance status.

2.10.3 Alert Management

o Alerts generated by the Wazuh Manager are displayed in the dashboard, were users can sort, filter, and investigate them. You can drill down into specific alerts to analyze the details of a potential security incident.

2.10.4 Threat Detection and Responses

o The Wazuh Dashboard provides a central platform to respond to security incidents, such as monitoring active responses or configuring automated responses to specific threats.

2.10.5 Compliance Monitoring

o Wazuh helps organizations comply with various regulatory frameworks (such as PCI DSS, GDPR, and HIPAA). The dashboard offers predefined compliance reports and tools to help monitor compliance status across the infrastructure.

2.10.6 File Integration Monitoring (FIM)

o The dashboard displays file changes in real-time, helping monitor critical system files and directories for any unauthorized modifications, which may indicate a security breach.

2.10.7 Vulnerability Detection

o With integrated vulnerability detection features, the Wazuh Dashboard helps users identify vulnerabilities in software packages, operating systems, and network configurations across their infrastructure.

2.10.8 Active Threat Hunting

o It allows security teams to actively hunt for potential threats or indicators of compromise by querying data and logs from multiple endpoints.

2.10.9 Log Data and Analysis

o Through integration with Wazuh Indexer (formerly Elasticsearch), the dashboard provides advanced search functionality, allowing users to perform full text searches logs, filter data, and analyze specific events.

Connect your network devices (such as servers, endpoints, and firewalls) to the Wazuh SIEM (Security Information and Event Management) system, you need to configure the Wazuh agents on those devices, which will then report their log data and events to the Wazuh Manager, and Figure 2-6 shows key features of Wazuh.

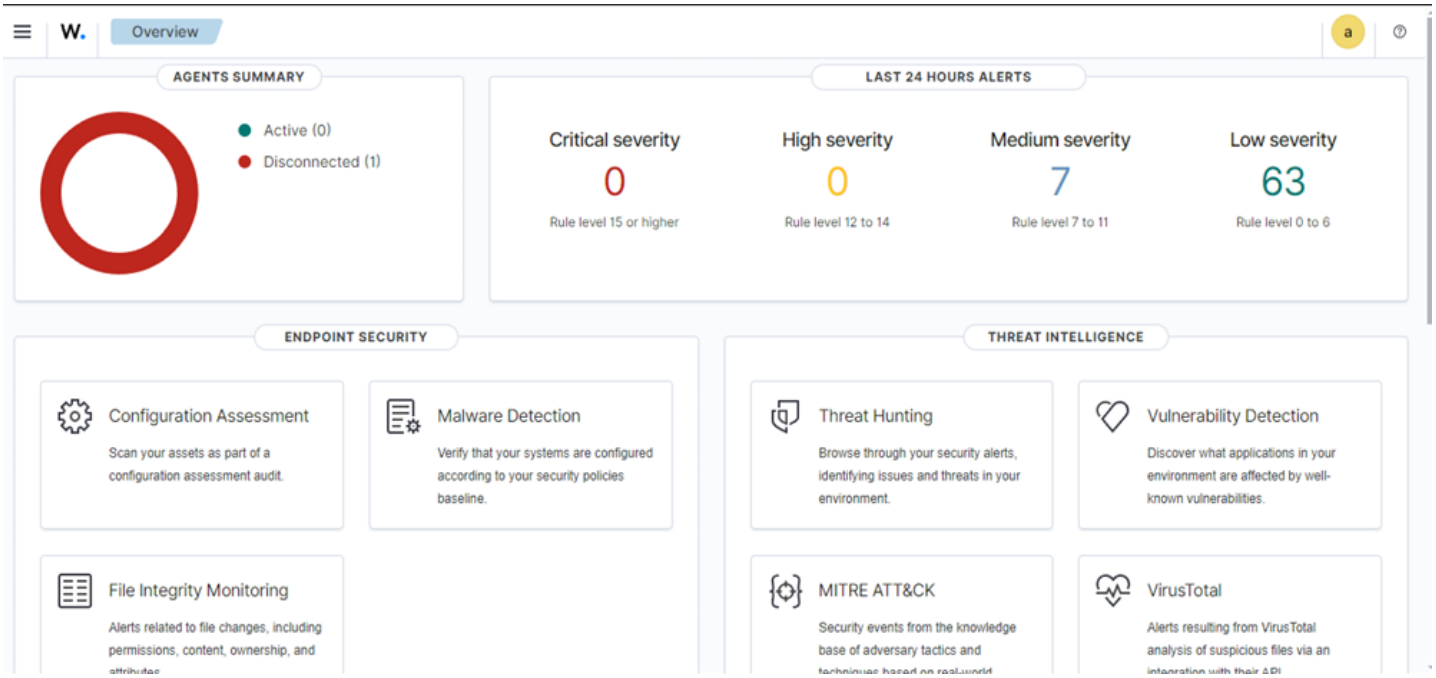


Fig [2-6] Shows Wazuh key Features and Functions.

2.10 How Wazuh-agent Works?

1- After installation, the Wazuh Agent continuously monitors the endpoint for various types of security-related activities, such as file changes, network activity, or log entries.

2- It sends the collected data to the Wazuh Manager using a secure communication protocol.

3- The Wazuh Manager processes and analyzes this data, generates alerts if necessary, and makes it available for visualization in the Wazuh Dashboard.

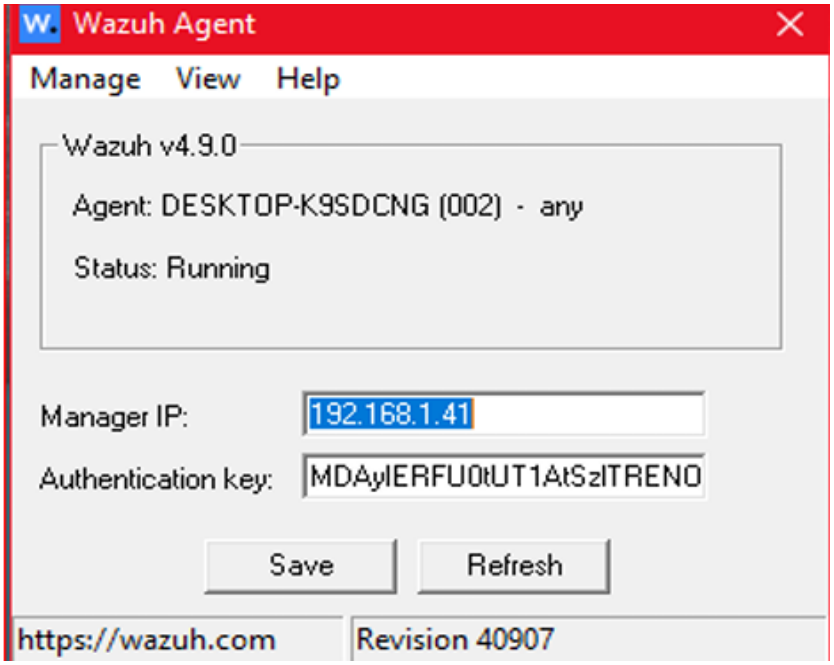


Fig [2-7] Shows Wazuh Agent Interface

Authentication key on the Wazuh Agent is a unique token used to securely register and authenticate the agent with the Wazuh Manager. This key is generated by the Wazuh Manager and is required to establish a trusted communication channel between the agent and the manager. The authentication key ensures that only authorized agents can connect to the Wazuh Manager and send logs or security data.

Key Points about the Authentication Key:

1. Purpose:

o The authentication key is used to securely register the Wazuh Agent with the Wazuh Manager.
o It helps prevent unauthorized agents from connecting to the Wazuh Manager.

2. One-Time Use:

o The key is a one-time-use token. After an agent has been registered using the authentication key, the key is no longer needed for subsequent communications.

3. Agent-Manager Communication:

o Once the agent is successfully registered, all communications between the agent and the Wazuh Manager occur over a secure channel (using TLS encryption), ensuring the confidentiality and integrity of the data transmitted.

4. Agent Registration Process:

o After installing the Wazuh Agent on a device, you need to register it with the Wazuh Manager by obtaining an authentication key from the manager.
o This key is generated using the manage_agents tool on the Wazuh Manager, and you copy it to the Wazuh Agent during the registration process.

To Get This Authentication Key Follow This Steps:

PuTTY is a free and open-source terminal emulator, network file transfer application, and SSH client. It is widely used for establishing secure, remote connections to servers and network devices, especially in environments running Linux, Unix, or other command line-based operating systems.

Key Features of PuTTY:

1. SSH Client:

o PuTTY primarily serves as an SSH (Secure Shell) client. SSH is a protocol used to securely connect to remote machines over an encrypted channel. This is commonly used by system administrators to manage remote servers and devices.

2. Telnet Client:

o In addition to SSH, PuTTY can also be used as a Telnet client, which is an older, less secure protocol for remote connections. SSH is preferred due to its encryption.

3. Serial Console:

o PuTTY can be used to connect to serial ports, making it useful for configuring network devices such as switches, routers, and firewalls that require console access.

4. Network File Transfers:

o PuTTY supports SCP (Secure Copy Protocol) and SFTP (SSH File Transfer Protocol) for transferring files securely between local and remote machines.

5. Customizable Interface:

o PuTTY provides options to customize the appearance of the terminal, such as changing font size, color schemes, and window behavior.

6. Port Forwarding:

o PuTTY supports SSH tunneling or port forwarding, allowing users to securely route traffic from one network port to another over an SSH connection.

7. Cross-Platform:

o While originally designed for Windows, PuTTY is also available on Linux and macOS. However, many Linux users prefer native tools like ssh for terminal access, but PuTTY is still popular for Windows users.

Common Use Cases:

1. Remote Server Administration:

o PuTTY is widely used by system administrators to connect to and manage remote Linux or Unix servers from their local machines. SSH provides a secure connection, allowing administrators to execute commands and perform configurations.

2. Network Device Configuration:

o PuTTY is used to connect to network devices like routers, switches, and firewalls via SSH or a serial interface for initial setup and configuration.

3. File Transfers:

o PuTTY's support for SCP and SFTP allows secure file transfers between machines, commonly used in remote file management or system backups.

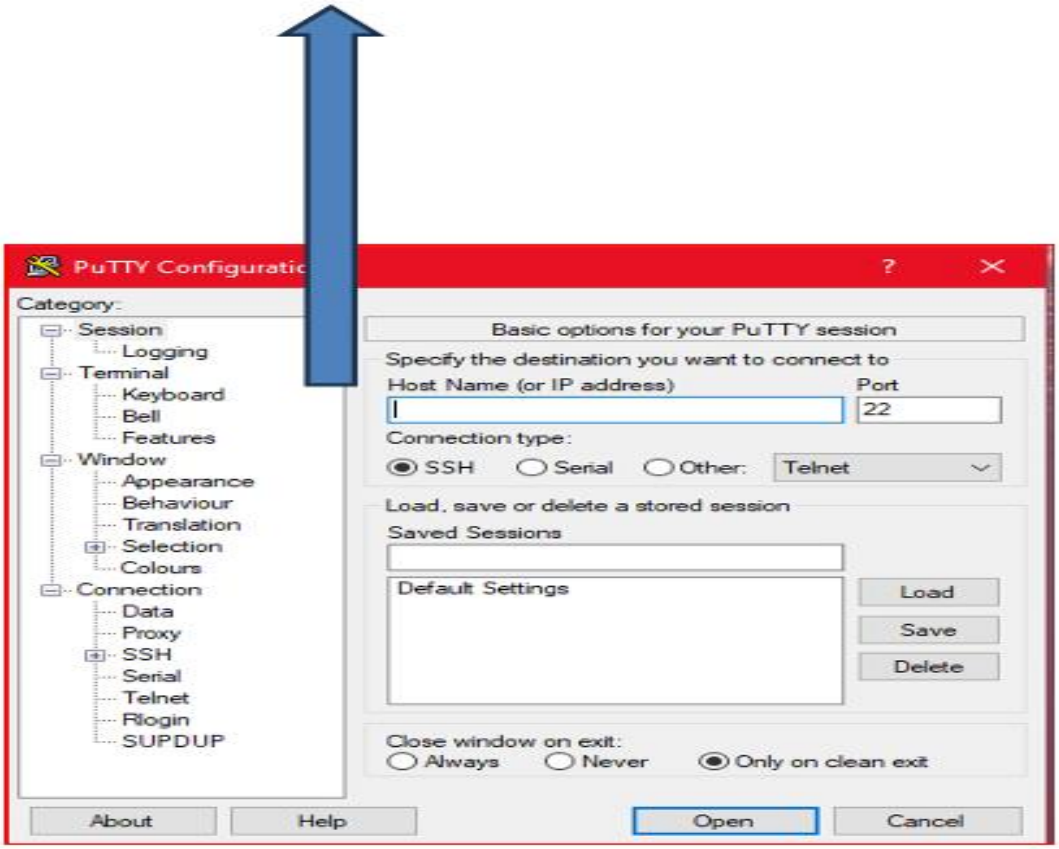
4. Secure Tunneling:

o PuTTY’s port forwarding feature is often used to create secure tunnels over SSH, enabling secure connections to services that are otherwise not encrypted (e.g., accessing a database through an SSH tunnel).

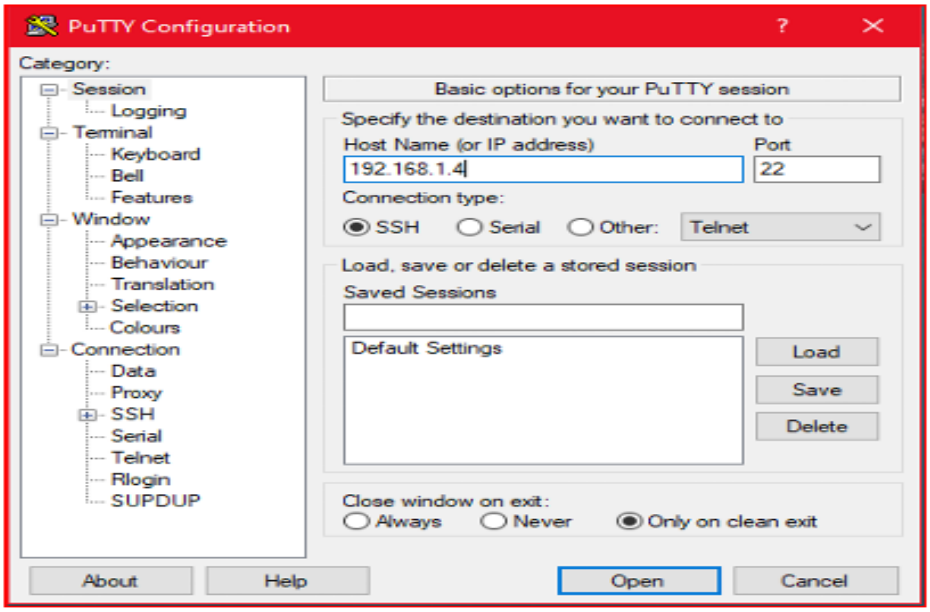
2.11 How to use putty?

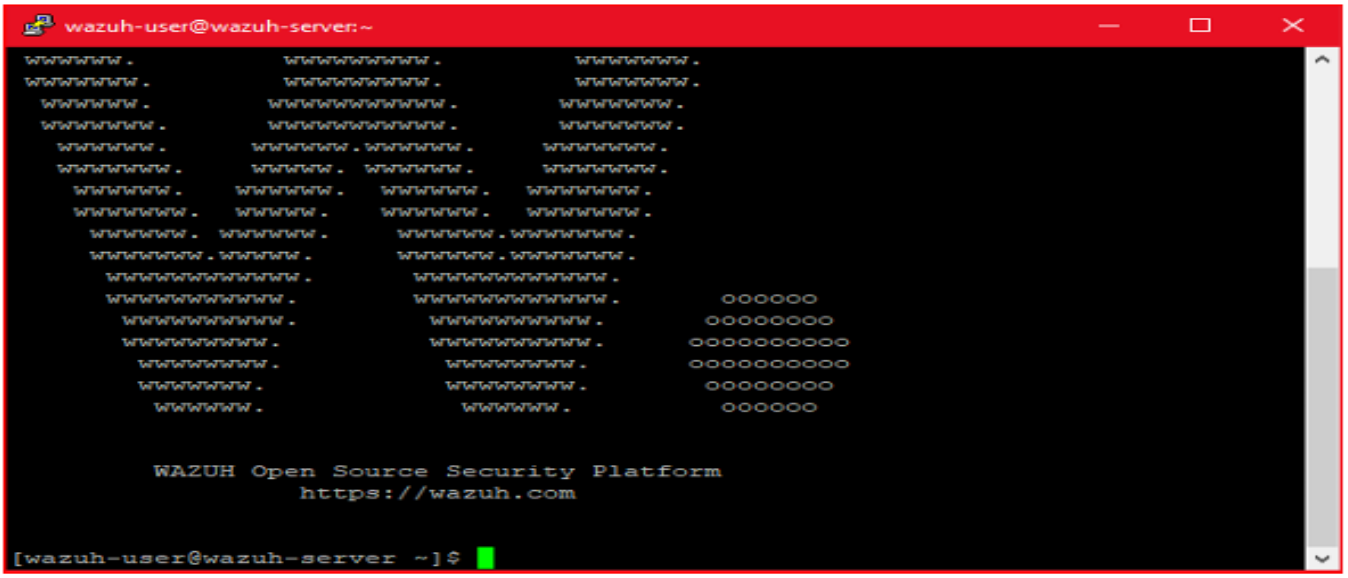
1- Open Putty and Choose SSH as Connection type.

You Need To Fill The ip Of Wazuh (OVA Server)



2- Write Server IP and connect to Wazuh-server.

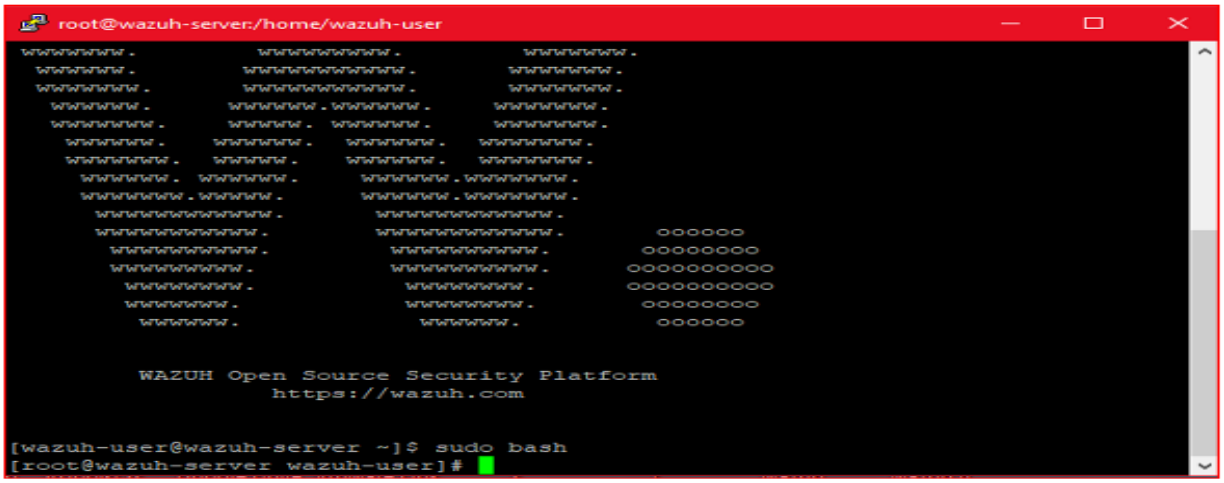




Username : wazuh-user

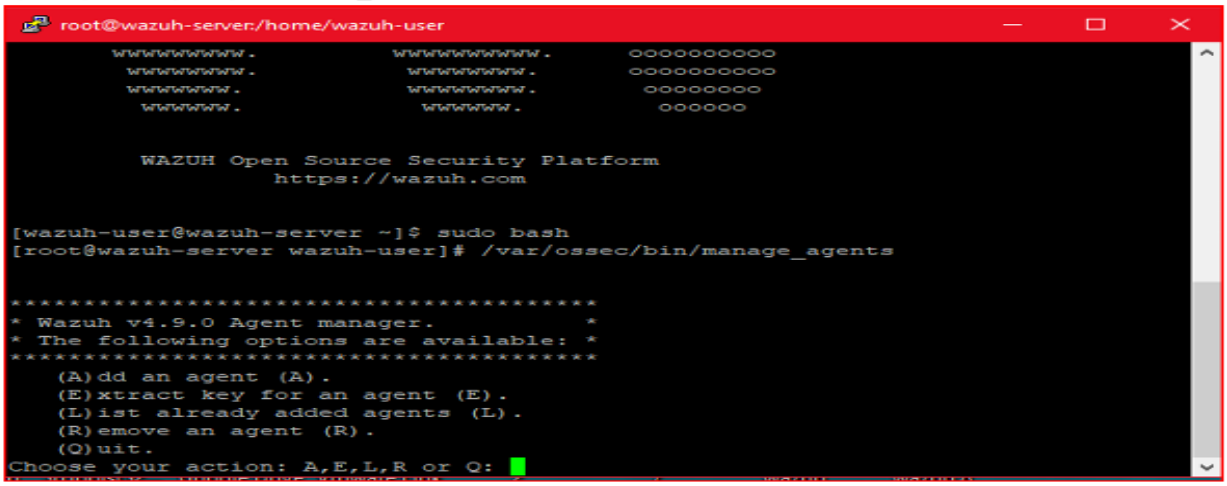
Password: wazuh

3- Write “sudo bash” to start writing up commands in server to add your agent.



We User sudo bash to get into root

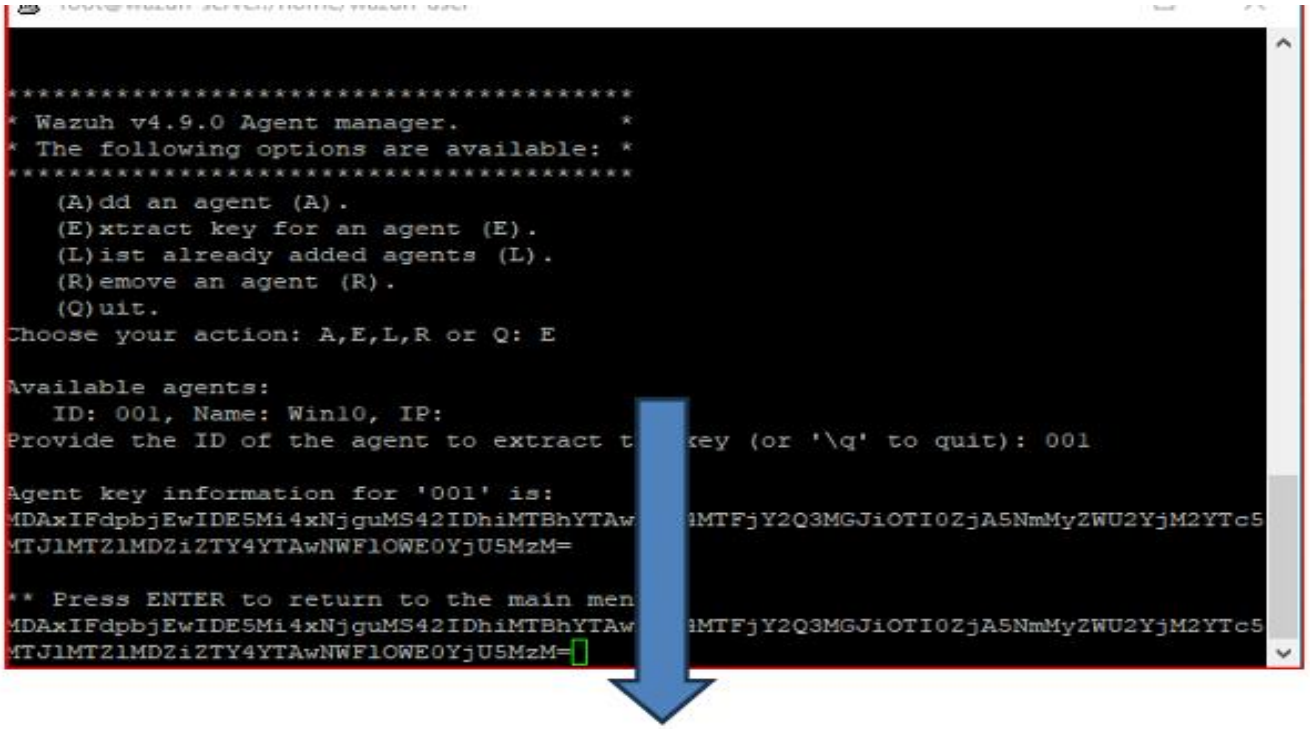
/VAR/OSSEC/BIN/MANAGE_AGENTS



To Get Connet With Wazuh Agent / We Choose A And named it with your name prefer and with the ip of your ipv4 address

---

4- Get Authentication Key to your added Agent.



- **To get the ip of id:001 we must go first to command prompt to get our ip :**

2.12 Monitoring and Alerting Setup Report

Introduction

- Overview: This section provides a brief introduction to the purpose of the monitoring and alerting setup. It highlights why monitoring is essential for the system's performance, security, and uptime.

- Scope: Define what areas or components the monitoring will cover (e.g., servers, network, applications, security, databases).

Objectives:

o Ensure system performance and availability.
o Detect anomalies, incidents, or failures.
o Enable real-time or near-real-time alerting for critical events.
o Maintain logs for future analysis and auditing.

System Environment Overview

Infrastructure: Describe the infrastructure being monitored (e.g., cloud, on-premises, hybrid) and the primary technologies used.
o Example: Linux-based servers, cloud VMs, containerized applications, etc.

Key Components: List the critical components to be monitored:
o Servers (Operating Systems)
o Databases (MySQL, MongoDB)
o Web Services (Apache, NGINX)
o Network (Firewalls, Routers, Switches)
o Applications
o Security Systems (SIEM, IDS/IPS)

Monitoring Tools

Selected Tools: Outline the tools chosen for monitoring.
o Wazuh: For security monitoring and log management.
o Prometheus: For system performance monitoring (CPU, memory, disk, network).
o Grafana: For visualizing metrics and logs from multiple sources.
o Nagios: For service availability monitoring and alerting.

Monitoring Setup

- Metrics Collection: Define the types of metrics collected:

o Performance Metrics:

- CPU, Memory, Disk Usage, Network Traffic.

o Application Metrics:
- HTTP requests, response times, database query times.

o Security Metrics:
- Unauthorized access attempts, file integrity, firewall breaches.

Log Collection:

o Centralized log collection through Wazuh.
o Log sources: system logs, security logs, application logs.

Alerting Setup

Alert Types:
o Critical: System or application downtime, security breaches, disk full.
o Warning: CPU usage above 90%, application response time degraded.
o Info: Daily status checks, backup completed successfully.

Thresholds:
o Define thresholds for each metric that trigger alerts.
o Example: CPU > 90% for 5 minutes triggers a warning alert.

Alert Channels:
o Email: For critical alerts to system administrators.
o SMS: For urgent security or system downtime alerts.
o Slack/Teams: For team collaboration and incident response.

Escalation Policies:
o Define the workflow for escalating critical issues (e.g., from Level 1 support to Level 2 and 3).
o Example: If the issue is not resolved in 15 minutes, notify the senior team.

Visualization and Dashboards

Grafana Dashboards:
o Real-time metrics for system performance.
o Security dashboards for alerts and incidents from Wazuh.
o Custom dashboards for network traffic, database performance, etc.

Dashboard Examples:
o CPU usage trends over time.
o Network traffic spikes.
o Real-time security incident logs.

Integration with Incident Management

Ticketing System Integration: Integration with systems like Jira, ServiceNow, or custom ticketing systems for automated incident creation.

Incident Response Workflow:
o Alerts automatically create tickets.
o Assigned personnel are notified.
o Ticket resolution tracked within the system.

Post-Incident Reporting:
o Automatically generate reports summarizing the incident, response, and resolution.

Security Monitoring and Compliance

Intrusion Detection:
o Using Wazuh to monitor for unauthorized access attempts and suspicious behavior.

Compliance Monitoring:
o Log and audit trails for compliance with regulations (e.g., GDPR, PCI DSS).

File Integrity Monitoring:
o Detect and alert changes to critical system files.

Testing and Validation

Regular Testing: How often tests are performed on the monitoring and alerting system to ensure they work as expected (e.g., monthly).

Simulated Incidents: Conduct periodic simulated incidents to verify alerting and response workflows.

Threshold Fine-Tuning: Adjust thresholds based on system behavior over time.

Future Improvements

Automation: Plans for automating incident response through scripts or machine learning.

Scalability: Plans for scaling the monitoring system as infrastructure grows.

Advanced Analytics: Use of predictive analytics or machine learning to detect anomalies early.

2.13 *Wazuh's Alerting Configuration & Integration With Virus Total*

To configure Wazuh to monitor near real-time changes in the ***C:\Users\<USER_NAME>\Downloads*** directory to check any file installed in downloads, follow these steps. This process includes installing necessary packages and creating an active response script to remove malicious files:

1. Locate the <syscheck> block in the Wazuh agent configuration file, typically found at ***C:\Program Files (x86)\ossec-agent\ossec.conf.***

2. Ensure that the ***<disabled> tag within the <syscheck> block is set to no***. This step activates the Wazuh File Integrity Monitoring (FIM) module, allowing it to track directory changes.

3. Add an entry inside the <syscheck> block to specify the directory you want to monitor in near real-time. For this use case, configure Wazuh to monitor the ***C:\Users\<USER_NAME>\Downloads*** directory by replacing ***<USER_NAME>*** with the appropriate username

> ***Note***
>
> Username of your windows client you can get it from "whoami" command and take the second partition after slash symbol for example: -
>
> ```
> C:\Users\dell>whoami
> desktop-1gfugr7\dell
> ```
>
> The username of this client is: dell, so directory of downloads will be in this path C:\Users\dell\Downloads

4. Download the Python executable installer from the official Python website here , After the download is complete, run the Python installer. Be sure to check the following options:

- **Install launcher for all users**
- **Add Python 3.X to PATH** (This adds the interpreter to the system's execution path)

Once the installation is finished, open PowerShell as an administrator and use pip to install PyInstaller.

```
> pip install pyinstaller
> pyinstaller --version
```

5. PyInstaller is used here to convert the remove-threat.py active response Python script into a standalone executable application that can be run on a Windows endpoint.

6. To convert the remove-threat.py script into a Windows executable, open PowerShell as an administrator and run the following command:

```
> pyinstaller -F \path_to_remove-threat.py
```

7. Transfer the remove-threat.exe executable file to the C:\Program Files (x86)\ossec-agent\active-response\bin directory.

8. To apply the changes, restart the Wazuh agent by running the following command in PowerShell as an administrator:

```
> Restart-Service -Name wazuh
```

9. After finishing your configuration in Wazuh-agent , now go to Wazuh-server after connecting on it from Putty.

Follow these steps on the Wazuh server to set up ***VirusTotal*** integration. This process will also enable and execute the active response script whenever a suspicious file is detected:

10. Add the following configuration to the ***/var/ossec/etc/ossec.conf*** file on the Wazuh server to enable VirusTotal integration. Replace ***<YOUR_VIRUS_TOTAL_API_KEY> with your VirusTotal API key***. This configuration will trigger a VirusTotal query whenever any of the rules in the FIM syscheck group are activated:

```
<ossec_config>
  <integration>
    <name>virustotal</name>
    <api_key><YOUR_VIRUS_TOTAL_API_KEY></api_key> <!-- Replace with your VirusTotal API key -->
    <group>syscheck</group>
    <alert_format>json</alert_format>
  </integration>
</ossec_config>
```

Note

You Should create an account in VirusTotal and after that click in your username in navbar and navigate to API key and get your API Key from here , and note that ***The free VirusTotal API rate limits requests to four per minute***. If you have a ***premium VirusTotal API key***, with a high frequency of queries allowed, you can ***add more rules besides these two. You can configure Wazuh to monitor more directories besides C:\Users\<USER_NAME>\Downloads***.

11. Append the following blocks to the Wazuh server ***/var/ossec/etc/ossec.conf*** file. This enables active response and triggers the remove-threat.sh script when VirusTotal flags a file as malicious:

```
<ossec_config>
  <command>
    <name>remove-threat</name>
    <executable>remove-threat.sh</executable>
    <timeout_allowed>no</timeout_allowed>
  </command>

  <active-response>
    <disabled>no</disabled>
    <command>remove-threat</command>
    <location>local</location>
    <rules_id>87105</rules_id>
  </active-response>
</ossec_config>
```

Note

Local Indicates that the file location of remove-threat and the malicious files and action will be loaded locally in the device which configured to be Wazuh-agent, and rule id: 87105 used for VirusTotal integration.

12. Add the following rules to the Wazuh server ***/var/ossec/etc/rules/local_rules.xml*** file to alert about the active response results.

```
<group name="virustotal,">
  <rule id="100092" level="12">
     <if_sid>657</if_sid>
     <match>Successfully removed threat</match>
     <description>$(parameters.program) removed threat located at
$(parameters.alert.data.virustotal.source.file)</description>
  </rule>

  <rule id="100093" level="12">
    <if_sid>657</if_sid>
    <match>Error removing threat</match>
    <description>Error removing threat located at
$(parameters.alert.data.virustotal.source.file)</description>
  </rule>
</group>
```

Level 12: OSSEC uses a **level system** to categorize the severity of alerts (from 0 to 15). A level of 12 indicates a relatively high-severity alert, suggesting that the rule relates to a significant event, such as the successful removal of a threat and level 15 used for data exfiltration cases.

if_sid (If Signature ID): This line specifies that this rule should only be evaluated if a previous alert with sid (signature ID) 657 has been triggered.

➔This means that **rule 100092** will only be considered if a previous event related

to rule 657 has occurred. The 657 might represent a rule that detects a specific threat or triggers a VirusTotal check.

Match Condition: This specifies the condition that triggers the rule. The rule will be activated when the text **"Successfully removed threat"** appears in the alert message or error message appeared.

$(parameters.program): This placeholder likely refers to the program or script that executed the threat removal.
$(parameters.alert.data.virustotal.source.file): This placeholder refers to the location or name of the file that was flagged by VirusTotal and subsequently removed.
Purpose: The description will contain information about which program removed the threat and the specific file that was addressed. It makes the alert message more informative for the user or analyst monitoring OSSEC.

Rule 100092: Triggered when a threat is successfully removed after a VirusTotal scan.
Rule 100093: Triggered when an error occurs during the attempt to remove a threat flagged by VirusTotal.

13. Restart the Wazuh manager to apply the configuration changes.

14. Start your **Attack emulation** to test your configuration.

15. Turn off real-time protection in the Windows Security app and then download an EICAR test file to the C:\Users\<USER_NAME>\Downloads folder by running following commands.

```
> Invoke-WebRequest -Uri https://secure.eicar.org/eicar.com.txt -OutFile eicar.txt
> cp .\eicar.txt C:\Users\<USER_NAME>\Downloads
```

16. Check the results of your ***near real-time anti-malware*** as shown in Figure 2-7.

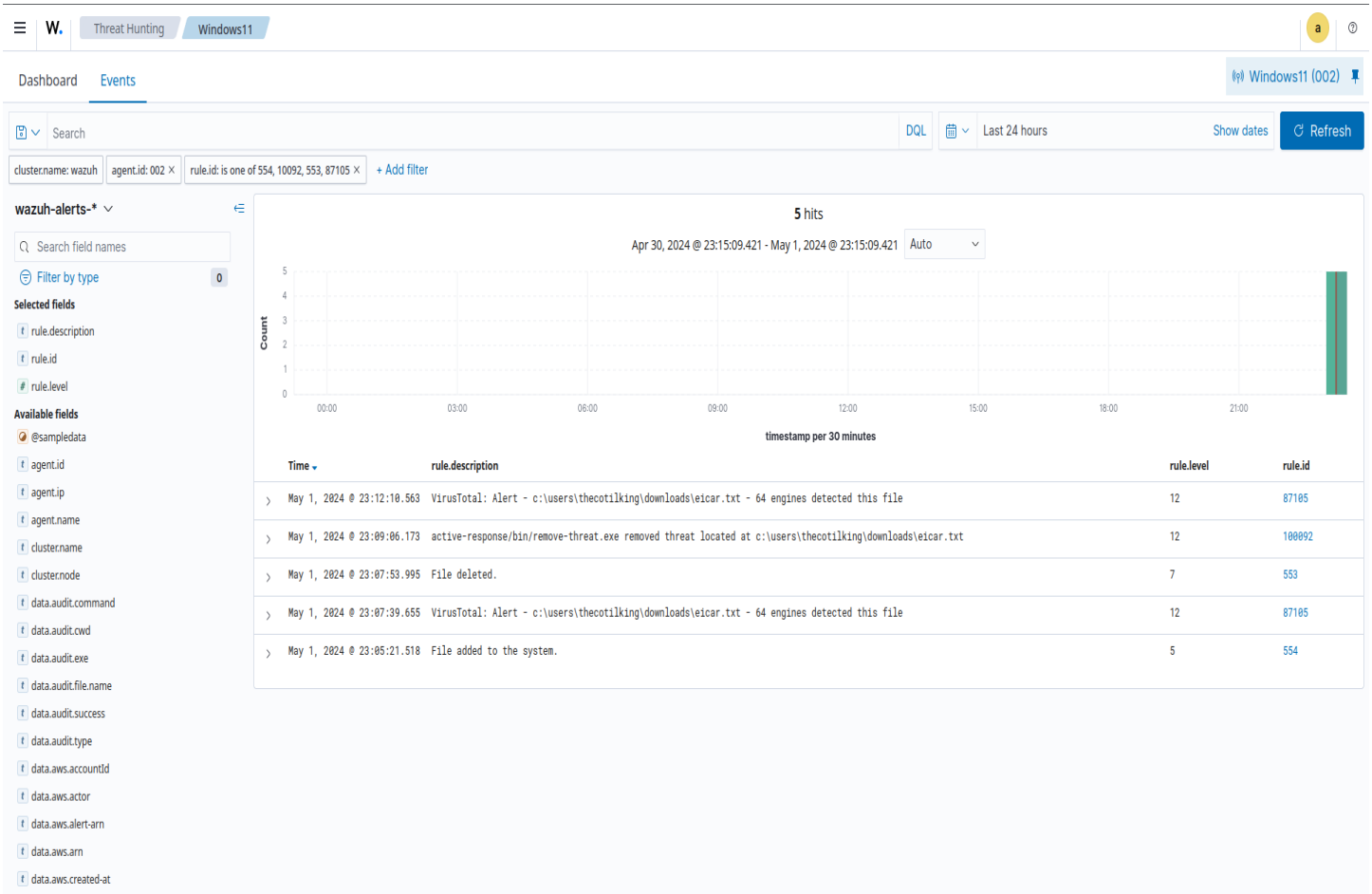


Fig [2-7] Shows the Alerts that come from malicious file downloaded in your Wazuh-agent.

Note

Now you can ask how malicious file has been removed from your Wazuh-agent or what's the source code of remove-threat.py or how it been implemented in python, all these concerns and question will be discussed in chapter 3 on what's our malware prevention strategy.

Chapter 3: Prevention Strategy and Training

3.1 Malware prevention strategy

3.1.1 General Prevention Strategies

A multi-layered defense strategy offers robust protection against different stages of the malware lifecycle by implementing various security measures across endpoint, network, email, web, and data protection domains. Here's a detailed explanation of each component:

1. Endpoint Protection

Endpoints are often the primary target for malware. Ensuring robust endpoint protection helps to defend against direct attacks and compromises.

- **Antivirus and Anti-malware Software**: Deploying reputable antivirus software on all endpoints helps detect and remove known malware. Regularly updating the virus definitions ensures the software can recognize the latest threats, while heuristic scanning allows for detecting unknown or new malware based on suspicious behavior patterns.

- **Application Whitelisting**: This approach allows only pre-approved applications to execute on endpoints. It prevents unauthorized software from running, which can significantly reduce the risk of malware execution. By using policies that define the allowed applications, suspicious software can be blocked before it has a chance to infect the system.

- **Patch Management**: Ensuring that operating systems, applications, and firmware are kept up to date is crucial for closing security gaps. An automated patch management system can prioritize critical updates and patch known vulnerabilities (like those identified by CVE numbers) that malware often exploits. It helps to limit the attack surface by eliminating software bugs that could be leveraged by attackers.

- **User Account Control (UAC)**: Enforcing the principle of least privilege limits the ability of users and applications to perform tasks that could compromise the system. For instance, users should only have the necessary permissions to perform their job functions, preventing malware from gaining administrative access to execute harmful actions or spread within the network.

2. Network Security

Network security controls protect against unauthorized access, lateral movement, and data exfiltration.

- **Firewall Configuration**: Firewalls serve as a frontline defense, controlling traffic based on predetermined security rules. By blocking unauthorized connections and restricting traffic to trusted IP addresses, ports, and services, firewalls help to prevent external threats from infiltrating the network. Stateful inspection allows the firewall to monitor active connections and decide whether to allow or deny traffic based on the state of the connection, while intrusion prevention capabilities can detect and block malicious traffic.

- **Network Segmentation**: Segmenting the network into different zones (e.g., DMZ, internal network, secure zones) helps contain potential threats and minimizes the damage in case of a breach. For example, placing sensitive data in a secure zone with stricter controls, while using ACLs to restrict access between zones, can prevent attackers from easily moving between network areas.

- **Intrusion Detection/Prevention Systems (IDS/IPS)**: IDS/IPS solutions monitor network traffic for indications of malicious activity, such as known attack signatures or deviations from normal traffic patterns (anomalies). While IDS detects and alerts about potential threats, IPS can take automated actions to block or contain suspicious traffic,
- thus, reducing the risk of successful attacks.

3. Email Security

Email remains a major vector for delivering phishing attacks and malware, making email security essential.

- **Email Filtering**: Email filtering solutions block spam, phishing attempts, and emails containing malicious attachments or links. This reduces the likelihood that a user will accidentally open a malicious email or click on a dangerous link.

- **Attachment Sandboxing**: This technology executes email attachments in a controlled environment (sandbox) to determine if they behave maliciously before being delivered to the recipient. If malicious behavior is detected, the attachment is quarantined or blocked.

- **Email Authentication Protocols**: Enforcing email authentication protocols like DMARC, SPF, and DKIM helps prevent email spoofing and ensures that the email sender is legitimate. These protocols verify that incoming emails are from trusted sources and have not been altered during transit, reducing the chances of successful phishing attacks.

4. Web Security

Web-based threats such as drive-by downloads, malicious websites, and phishing can be mitigated with web security measures.

- **Web Filtering**: Web filtering solutions are crucial for restricting access to high-risk websites, reducing exposure to online threats such as malware and phishing. To further enhance security, advanced backend techniques can be employed, including application-level guard controls, secure password hashing with salting (using algorithms like bcrypt or Argon2), and robust encryption using AES-256 with custom modifications. These modifications can involve custom key derivation functions, cipher mode enhancements, key rotation, and splitting to make decryption nearly impossible for attackers. Additionally, code obfuscation and integrity checks can protect encryption routines from reverse-engineering, creating a multi-layered defense that goes beyond conventional measures to safeguard sensitive data and web applications.

- **SSL/TLS Inspection**: As a significant amount of web traffic is encrypted, it is important to inspect SSL/TLS traffic to identify hidden threats. SSL/TLS inspection decrypts traffic for inspection, allowing security tools to detect and block malware concealed in encrypted web traffic.

- **Content Delivery Network (CDN)**: Using a CDN for web application delivery not only improves performance but also provides built-in security features like DDoS protection and a web application firewall (WAF). The WAF helps protect web applications from common attacks such as SQL injection, cross-site scripting (XSS), and other vulnerabilities.

5. Data Protection and Encryption

Protecting data is critical for reducing the impact of data breaches and preventing data loss.

- **Data Loss Prevention (DLP)**: DLP solutions monitor and control the movement of sensitive data, preventing unauthorized data transfers or leakage. For instance, DLP can block or log attempts to send confidential files outside the organization, ensuring data security policies are enforced.

- **Encryption**: Encrypting data at rest and in transit ensures that even if data is intercepted or stolen, it cannot be easily accessed or used by unauthorized individuals. Using strong encryption algorithms, such as AES-256, makes it difficult for attackers to decrypt the data.

- **Access Control Policies**: Enforcing strict access control policies limits who can access sensitive data. Policies can be based on roles and responsibilities, ensuring that only authorized users can access certain data. Role-based access control (RBAC) and multi-factor authentication (MFA) are effective measures to further secure sensitive information.

Each layer in this defense strategy complements the others, making it harder for malware to achieve its objectives. By integrating these measures, organizations can reduce risks, detect threats earlier, and respond more effectively to potential incidents.

3.1.2 Wazuh's exe file used for prevention strategy (remove-threat.py)

Do you remember remove-threat.py which we change it to exe to run it with defined VirusTotal rules to add alerts to our client logs and remove malicious file and we explain this part step-by-step in chapter 2 and you can review it here , now we will explain every part in this code.

```
import os
import sys
import json
import datetime

if os.name == 'nt':
    LOG_FILE = "C:\\Program Files (x86) \\ossec-agent\\active-response\\active-responses.log"
else:
    LOG_FILE = "/var/ossec/logs/active-responses.log"

ADD_COMMAND = 0
DELETE_COMMAND = 1
CONTINUE_COMMAND = 2
ABORT_COMMAND = 3

OS_SUCCESS = 0
OS_INVALID = -1
```

```
class message:
    def __init__(self):
        self.alert = ""
        self.command = 0

def write_debug_file(ar_name, msg):
    with open(LOG_FILE, mode="a") as log_file:
        log_file.write(str(datetime.datetime.now().strftime('%Y/%m/%d %H:%M:%S')) + "
" + ar_name + ": " + msg +"\n")

def setup_and_check_message(argv):

    # get alert from stdin
    input_str = ""
    for line in sys.stdin:
        input_str = line
        break


    try:
        data = json.loads(input_str)
    except ValueError:
        write_debug_file(argv[0], 'Decoding JSON has failed, invalid input format')
        message.command = OS_INVALID
        return message

    message.alert = data

    command = data.get("command")

    if command == "add":
        message.command = ADD_COMMAND
    elif command == "delete":
        message.command = DELETE_COMMAND
    else:
        message.command = OS_INVALID
        write_debug_file(argv[0], 'Not valid command: ' + command)

    return message
```

```
def send_keys_and_check_message(argv, keys):

    # build and send message with keys
    keys_msg = json.dumps({"version": 1,"origin":{"name": argv[0],"module":"active-
response"},"command":"check_keys","parameters":{"keys":keys}})

    write_debug_file(argv[0], keys_msg)

    print(keys_msg)
    sys.stdout.flush()

    # read the response of previous message
    input_str = ""
    while True:
        line = sys.stdin.readline()
        if line:
            input_str = line
            break

    # write_debug_file(argv[0], input_str)

    try:
        data = json.loads(input_str)
    except ValueError:
        write_debug_file(argv[0], 'Decoding JSON has failed, invalid input format')
        return message

    action = data.get("command")

    if "continue" == action:
        ret = CONTINUE_COMMAND
    elif "abort" == action:
        ret = ABORT_COMMAND
    else:
        ret = OS_INVALID
        write_debug_file(argv[0], "Invalid value of 'command'")

    return ret
```

```
def main(argv):

    write_debug_file(argv[0], "Started")

    # validate json and get command
    msg = setup_and_check_message(argv)

    if msg.command < 0:
        sys.exit(OS_INVALID)

    if msg.command == ADD_COMMAND:
        alert = msg.alert["parameters"]["alert"]
        keys = [alert["rule"]["id"]]
        action = send_keys_and_check_message(argv, keys)

        # if necessary, abort execution
        if action != CONTINUE_COMMAND:

            if action == ABORT_COMMAND:
                write_debug_file(argv[0], "Aborted")
                sys.exit(OS_SUCCESS)
            else:
                write_debug_file(argv[0], "Invalid command")
                sys.exit(OS_INVALID)

        try:
            file_path =
msg.alert["parameters"]["alert"]["data"]["virustotal"]["source"]["file"]
            if os.path.exists(file_path):
                os.remove(file_path)
            write_debug_file(argv[0], json.dumps(msg.alert) + " Successfully removed
threat")
        except OSError as error:
            write_debug_file(argv[0], json.dumps(msg.alert) + "Error removing
threat")


    else:
        write_debug_file(argv[0], "Invalid command")

    write_debug_file(argv[0], "Ended")
```

```
    sys.exit(OS_SUCCESS)

if __name__ == "__main__":
    main(sys.argv)
```

This is the whole code, seems complicated! but let us explain it block by block: -

1- Imports and Global Variable Definitions

```
import os
import sys
import json
import datetime
if os.name == 'nt':
    LOG_FILE = "C:\\Program Files (x86) \\ossec-agent\\active-response\\active-responses.log"
else:
    LOG_FILE = "/var/ossec/logs/active-responses.log"

ADD_COMMAND = 0
DELETE_COMMAND = 1
CONTINUE_COMMAND = 2
ABORT_COMMAND = 3

OS_SUCCESS = 0
OS_INVALID = -1
```

This block imports the necessary modules:

- **os** is used for operating system-dependent functionalities.
- **sys** allows interaction with the system, like reading and writing data from standard input/output.
- **json** handles JSON encoding and decoding.

- **datetime** is used for timestamping log entries.
- This determines the location of the log file based on the operating system. For **Windows (nt)**, a different file path is used compared to Unix-based systems.
- These constants define commands and status codes used throughout the script:
- **ADD_COMMAND, DELETE_COMMAND**, etc., represent different command types.
- **OS_SUCCESS and OS_INVALID** indicate whether the operation succeeded or encountered an error.

2. Message Class Definition

```
class message:
    def __init__(self):
        self.alert = ""
        self.command = 0
```

The message class defines a structure to store alert details and the associated command. It initializes alert as an empty string and command as zero.

3. Logging Function

```
def write_debug_file(ar_name, msg):
    with open(LOG_FILE, mode="a") as log_file:
        log_file.write(str(datetime.datetime.now().strftime('%Y/%m/%d %H:%M:%S')) + "
" + ar_name + ": " + msg +"\n")
```

- opens the `LOG_FILE` in append mode (`"a"`), ensuring that new log entries are added to the end of the file without overwriting existing content.
- It writes a log entry that includes the current timestamp, the name of the script (`ar_name`), and the actual message (`msg`) provided as input.
- The timestamp is formatted as `YYYY/MM/DD HH:MM:SS` to ensure consistency.

4. Message Setup and Validation

```
def setup_and_check_message(argv):

    # get alert from stdin
    input_str = ""
    for line in sys.stdin:
        input_str = line
        break



    try:
        data = json.loads(input_str)
    except ValueError:
        write_debug_file(argv[0], 'Decoding JSON has failed, invalid input format')
        message.command = OS_INVALID
        return message

    message.alert = data

    command = data.get("command")

    if command == "add":
        message.command = ADD_COMMAND
    elif command == "delete":
        message.command = DELETE_COMMAND
    else:
        message.command = OS_INVALID
        write_debug_file(argv[0], 'Not valid command: ' + command)

    return message.
```

This function reads and validates the incoming message:

Reading Input: It reads the input from standard input (`sys.stdin`) line by line, expecting a JSON-formatted string containing alert information.

JSON Decoding: It attempts to parse the input string using `json.loads()`. If parsing fails (due to an invalid JSON format), it logs an error and sets the `command` to `OS_INVALID`.

Command Checking: If parsing is successful, it checks the `"command"` field in the JSON data:

- If the command is `"add"`, it sets `message.command` to `ADD_COMMAND`.
- If the command is `"delete"`, it sets `message.command` to `DELETE_COMMAND`.
- If the command is neither of these, it logs an invalid command message and sets the command to `OS_INVALID`.

5. Sending and Validating Keys

```
def send_keys_and_check_message (argv, keys):

    # build and send message with keys
    keys_msg = json.dumps({"version": 1,"origin":{"name": argv[0],"module":"active-
response"},"command":"check_keys","parameters":{"keys":keys}})

    write_debug_file(argv[0], keys_msg)

    print(keys_msg)
    sys.stdout.flush()

    # read the response of previous message
    input_str = ""
    while True:
        line = sys.stdin.readline()
        if line:
            input_str = line
            break

    # write_debug_file(argv[0], input_str)

    try:
        data = json.loads(input_str)
    except ValueError:
        write_debug_file(argv[0], 'Decoding JSON has failed, invalid input format')
        return message

    action = data.get("command")

    if "continue" == action:
        ret = CONTINUE_COMMAND
    elif "abort" == action:
        ret = ABORT_COMMAND
    else:
        ret = OS_INVALID
```

```
        write_debug_file(argv[0], "Invalid value of 'command'")

    return ret
```

This function builds and sends a message to validate keys:

Building the Message: It creates a JSON message containing details about the origin, command, and parameters. The `keys` parameter is the list of keys to be validated.

Logging and Sending: It logs the message to the debug file and sends it via `stdout`.

Reading the Response: The function waits for a response by reading from `stdin`. It expects a valid JSON response.

Decoding the Response: It tries to parse the response JSON. If parsing fails, it logs an error.

Validating the Command in Response: It checks the `"command"` field in the response:

- If the command is `"continue"`, it returns `CONTINUE_COMMAND`.
- If the command is `"abort"`, it returns `ABORT_COMMAND`.
- If neither, it logs an invalid command error and returns `OS_INVALID`.

6. Main Function for Execution

```
def main(argv):

    write_debug_file(argv[0], "Started")

    # validate json and get command
    msg = setup_and_check_message(argv)

    if msg.command < 0:
        sys.exit(OS_INVALID)

    if msg.command == ADD_COMMAND:
        alert = msg.alert["parameters"]["alert"]
        keys = [alert["rule"]["id"]]
        action = send_keys_and_check_message(argv, keys)

        # if necessary, abort execution
        if action != CONTINUE_COMMAND:

            if action == ABORT_COMMAND:
                write_debug_file(argv[0], "Aborted")
                sys.exit(OS_SUCCESS)
            else:
                write_debug_file(argv[0], "Invalid command")
                sys.exit(OS_INVALID)

        try:
            file_path =
msg.alert["parameters"]["alert"]["data"]["virustotal"]["source"]["file"]
            if os.path.exists(file_path):
                os.remove(file_path)
            write_debug_file(argv[0], json.dumps(msg.alert) + " Successfully removed
threat")
        except OSError as error:
            write_debug_file(argv[0], json.dumps(msg.alert) + "Error removing
threat")


    else:
        write_debug_file(argv[0], "Invalid command")
```

```
    write_debug_file(argv[0], "Ended")

    sys.exit(OS_SUCCESS)
```

The main function coordinates the entire execution:

Start Logging: It logs "Started" at the beginning.

Message Validation: It calls setup_and_check_message() to validate the incoming message.

Command Execution: If the command is invalid, it exits. If the command is ADD_COMMAND, it retrieves the alert information and extracts keys for validation.

Key Validation: It calls send_keys_and_check_message() to validate the keys. If the response requires aborting, it logs "Aborted" and exits.

Threat Removal: If the command is valid and allowed to proceed, it attempts to remove a file specified in the alert. If the file is removed successfully, it logs the success; otherwise, it logs an error.

End Logging: It logs "Ended" at the end and exits with a success status.

7. Script Execution Entry Point

```
if __name__ == "__main__":
    main(sys.argv)
```

This ensures that the script runs the `main()` function when executed as a standalone program. It passes the command-line arguments to `main()`. This block allows the script to be used as a standalone executable or imported as a module without running the `main()` function automatically.

3.2 User awareness training

Creating user awareness training materials is essential for educating employees about cybersecurity threats and best practices. Here's a comprehensive documentation outline on developing user awareness training materials, covering various aspects such as the purpose, key topics, delivery methods, evaluation, and continuous improvement.

User Awareness Training Documentation

3.2.1 Introduction

User awareness training is a critical component of an organization's cybersecurity strategy. The purpose of this training is to educate employees about potential security threats, promote a security-conscious culture, and empower them to recognize and respond to security incidents effectively.

3.2.1.1 Objectives

- **Enhance Cybersecurity Awareness:** Increase understanding of common threats such as phishing, malware, and social engineering.
- **Promote Safe Practices:** Encourage safe computing habits, including password management and safe internet browsing.
- **Empower Employees:** Equip employees with the knowledge and tools needed to identify and report suspicious activities.

3.2.2 Key Topics to Cover

3.2.2.1 Understanding Cybersecurity Threats

- **Phishing Attacks:**
    - Definition and examples of phishing.
    - How to identify phishing emails and messages.

- Reporting procedures for suspected phishing attempts.

- **Malware:**
  - Overview of different types of malware (viruses, worms, ransomware).
  - Signs of malware infection (slow performance, unexpected behavior).
  - Safe downloading practices.
- **Social Engineering:**
  - Techniques used by attackers (pretexting, baiting, tailgating).
  - Real-world scenarios to illustrate social engineering attacks.

3.2.2.2 Safe Computing Practices

- **Password Management:**
  - Importance of strong, unique passwords.
  - Using password managers and enabling multi-factor authentication (MFA).
  - Periodic password updates and avoiding password reuse.
- **Safe Internet Browsing:**
  - Identifying secure websites (HTTPS vs. HTTP).
  - Avoiding suspicious links and downloads.
  - Understanding browser privacy settings and extensions.

3.2.2.3 Data Protection

- **Handling Sensitive Information:**
  - Identifying sensitive data (personal data, financial information).
  - Best practices for sharing sensitive information (encryption, secure channels).
  - Understanding the organization’s data protection policies.
- **Device Security:**
  - Importance of keeping software up to date (operating systems, applications).
  - Using antivirus software and firewalls.
  - Recognizing physical security threats (lost devices, unauthorized access).

3.2.2.4 Incident Reporting

- **Recognizing Security Incidents:**
  - Definition of security incidents and breaches.
  - Signs of a potential security incident.
- **Reporting Procedures:**
  - Clear steps for reporting incidents (who to contact, information to provide).
  - Importance of timely reporting and cooperation with the IT/security team.

3.2.3 Delivery Methods

3.2.3.1 Training Formats

- **In-Person Workshops:**
  - Interactive sessions with real-world scenarios and group discussions.
  - Opportunities for questions and answers.
- **Online Courses:**
  - Self-paced e-learning modules with quizzes and assessments.
  - Access to resources and materials for further reading.
- **Webinars:**
  - Live online sessions covering key topics.
  - Opportunities for interaction and engagement through Q&A.

3.2.3.2 Training Materials

- **Presentations:**
  - Slides summarizing key topics, complete with visuals and examples.
- **Handouts:**
  - Quick reference guides, checklists, and infographics summarizing key points.
- **Videos:**
  - Engaging video content demonstrating threats and responses.

- **Quizzes:**
    - Short assessments to gauge understanding and reinforce learning.

3.2.4 Evaluation and Feedback

3.2.4.1 Assessing Training Effectiveness

- **Pre- and Post-Training Assessments:**
    - Quizzes before and after training to measure knowledge gains.
- **Feedback Surveys:**
    - Collecting participant feedback on training content, delivery, and relevance.
- **Incident Tracking:**
    - Monitoring the number of reported incidents before and after training to assess impact.

3.2.4.2 Continuous Improvement

- **Regular Review of Content:**
    - Updating training materials to reflect emerging threats and changes in technology.
- **Follow-Up Training:**
    - Offering periodic refresher courses and updates to keep employees informed.
- **Engagement Metrics:**
    - Analyzing participation rates and feedback to improve future training sessions.

3.2.5 Conclusion

User awareness training is a vital part of an organization’s cybersecurity strategy. By educating employees about potential threats and safe computing practices, organizations can significantly reduce their risk of security incidents. Regular training, evaluation, and updates to the training program are essential to maintain a high level of security awareness among employees.

3.2.6 Avoid Phishing mail.

3.2.6.1 Avoiding Phishing Emails

Phishing emails are deceptive messages that trick recipients into revealing sensitive information or clicking on malicious links. To avoid falling for phishing, users should follow these best practices:

- **Examine the Sender's Email Address:**
  - Be cautious of emails from unfamiliar senders or email addresses that don’t match the company’s domain. Look out for slight misspellings or odd variations in familiar email addresses.

- **Beware of Suspicious Links and Attachments:**
  - Avoid clicking on links or downloading attachments from unsolicited emails. Hover over links to view the actual URL and ensure it matches the intended website.

- **Check for Generic Greetings and Unusual Language:**
  - Phishing emails often use generic greetings such as "Dear Customer" or contain grammatical errors. Look for personalized emails from trusted sources instead.

- **Look for Urgency and Threats:**
  - Phishing emails often use urgent language to scare users into taking quick action (e.g., "Your account will be locked in 24 hours"). If an email seems overly urgent or threatening, verify its legitimacy through another communication channel.

- **Verify Email Requests for Sensitive Information:**
    - Legitimate companies will not ask for sensitive information like passwords, credit card details, or personal identification through email. Always verify requests for such information via phone or official company contacts.

- **Enable Anti-Phishing Features:**
    - Make sure that your email client’s anti-phishing settings are enabled and utilize tools that help identify suspicious messages.

- **Use Multi-Factor Authentication (MFA):**
    - Even if you fall for a phishing fraud, MFA adds an extra layer of security that helps protect accounts from unauthorized access.

3.2.6.2 Reporting Phishing Emails to SOC Analysts

Having a clear reporting procedure ensures phishing attempts are swiftly investigated and mitigated. Here's how end users should report suspected phishing emails:

- **Use the Designated Reporting Tool:**
    - If your company uses a tool for reporting suspicious emails (such as a "Report Phishing" button in the email client), use it to instantly send the email to the security team.

- **Forward the Email to a SOC Analyst:**
    - If no automated tool is available, forward the suspected phishing email to the designated SOC email address (e.g., soc-team@company.com). Include a brief description of why you suspect the email is phishing.

- **Provide Detailed Information:**
    - When reporting, include relevant details such as the sender's email address, the subject line, and any actions taken (e.g., clicked links, opened attachments).

- **Follow Up:**
    - If you interacted with the phishing email (clicked a link, entered credentials, or opened an attachment), immediately notify the SOC team or IT department for further guidance.

- **Avoid Deleting the Email:**
    - Do not delete the email until instructed by the SOC team, as it may be needed for investigation.

3.2.6.3 SOC Analysts' Response Workflow

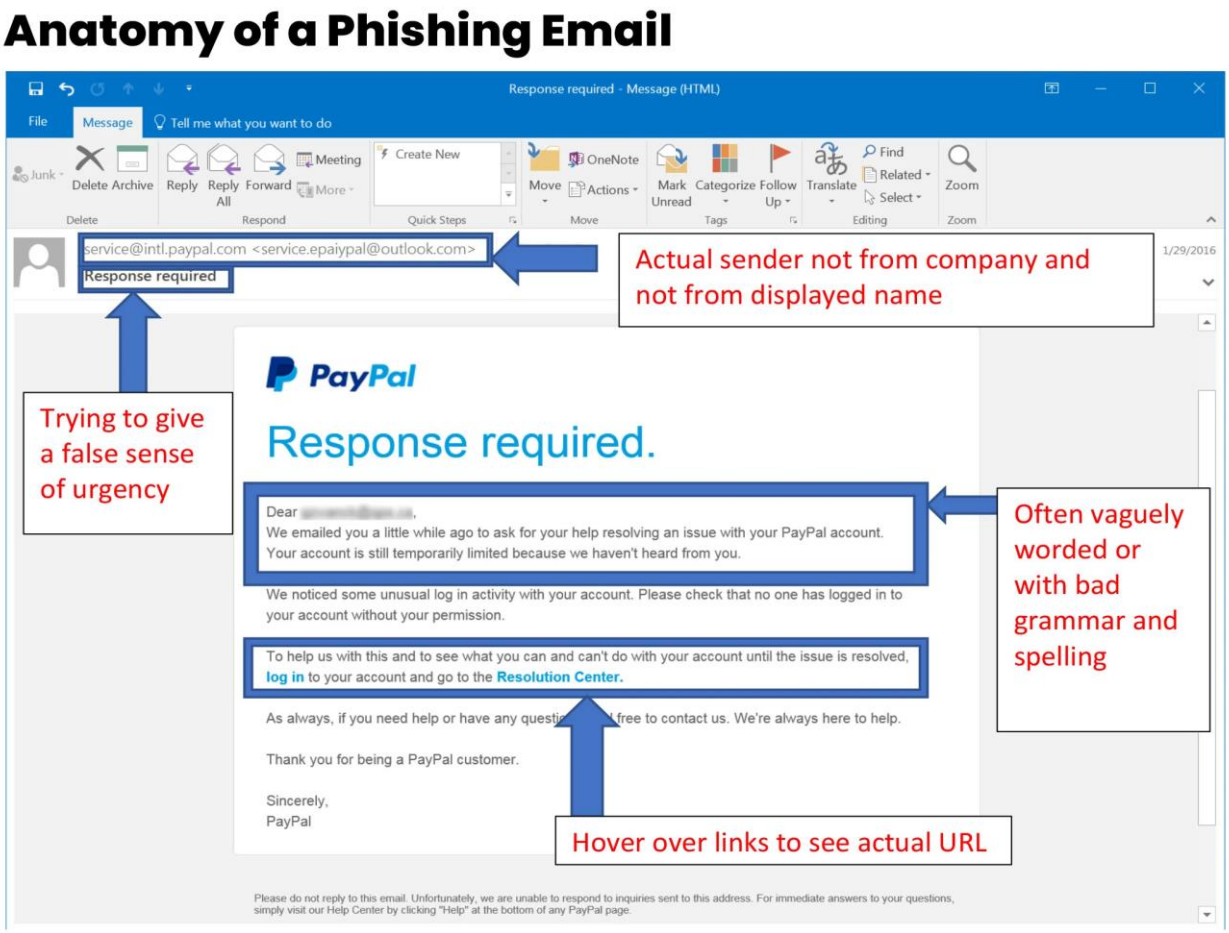
Once a phishing report is received, SOC analysts will typically follow these steps:

1. **Analyze the Email:**
    - Verify whether the reported email is a phishing attempt and assess its threat level.
2. **Contain and Eradicate the Threat:**
    - If the email contains malware or compromised links, SOC analysts will block the

source, quarantine affected systems, and alert users.

3. **Notify Affected Users:**
    - Provide guidance to any users who may have interacted with the phishing email.
4. **Incident Documentation:**
    - Log the incident details and response actions in the incident management system for future reference and improvement.
5. **Awareness Feedback:**
    - Use the incident as a learning opportunity by sharing details (without confidential information) to improve user training.

3.2.7 Phishing mail shapes



Even if you think the email is legitimate, if it is not something you are expecting it is a good idea to contact the person you believe to be the sender “out-of-band,” or by another method than clicking “reply” or any links in the email. For example, call a phone number you know belongs to the institution or person or go directly to their website by typing in the URL.

*original image taken from phishing.org

Google

Gmail

Important: Your Password will expire in 1 day(s)

Inbox x

MyUniversity 12:18 PM (50 minutes ago)

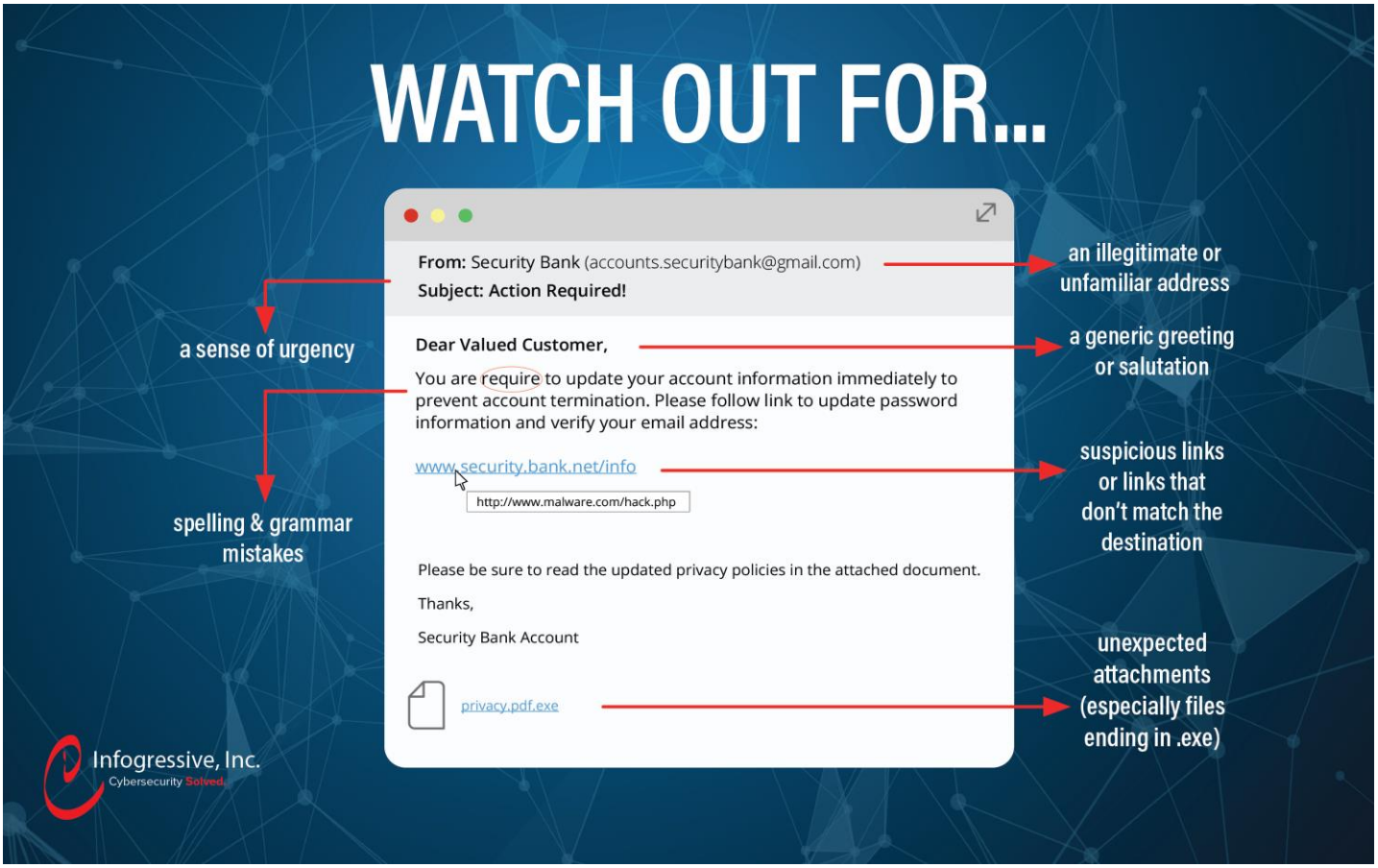
to me

Dear network user,

This email is meant to inform you that your MyUniversity network password will expire in 24 hours.
Please follow the link below to update your password
myuniversity.edu/renewal

MY UNIVERSITY

Thank you
MyUniversity Network Security Staff



Figures [3-1] Shows Phishing mail shape and anatomy.

3.2.8 Additional Resources

(146) Cyber Security Tips for End Users - YouTube

An End User's Guide to Cybersecurity Awareness (youtube.com)

Understanding the Importance of End-User Security (youtube.com)

Security Awareness for Office 365 (youtube.com)

(146) WATCH NOW!! Cyber Security Awareness Training For Employees & End Users - InfoSec Pat 2022 Video - YouTube