**Conducting Vulnerability Assessment and Penetration Testing on a Simulated Web Application Environment**

**Source Codes**:

**Commands on Kali Linx**

#!/bin/bash

# Source Code for Vulnerability Assessment & Penetration Testing Project

# Author: Sanjana S

# Date: 23 Oct 2025

# Set target IP addresses

TARGET\_IPS=("10.0.2.1" "10.0.2.2" "10.0.2.6" "10.0.2.15")

# Create output directory

OUT\_DIR=~/pentest\_outputs

mkdir -p $OUT\_DIR

# Loop through each target IP

for IP in "${TARGET\_IPS[@]}"; do

mkdir -p "$OUT\_DIR/$IP"

# Full TCP scan

nmap -Pn -sT -p- -T4 -oN "$OUT\_DIR/$IP/full\_tcp\_$IP.txt" $IP

# HTTP scan for ports 80 and 443

nmap -p 80,443 --script=http-title,http-headers,http-enum -oN "$OUT\_DIR/$IP/nmap\_http.txt" $IP

# Curl HTTP check

curl -I http://$IP:80 || curl -I http://$IP || curl -I https://$IP --insecure || echo "HTTP check failed"

# Scan specific ports and grab service banners

PORTS=(22 2289 35768 51078)

nmap -Pn -sV --script=banner -p ${PORTS[@]} -oN "$OUT\_DIR/$IP/ports\_banner.txt" $IP

# Grab banners with netcat

for PORT in "${PORTS[@]}"; do

timeout 3 bash -c "echo | nc -w 3 $IP $PORT" > "$OUT\_DIR/$IP/port${PORT}\_banner.txt" 2>/dev/null || true

done

done

# DNS check for 10.0.2.1

dig @10.0.2.1 any +noall +answer > "$OUT\_DIR/10.0.2.1/dig\_any.txt" 2>/dev/null || echo "no dns answer"

# Flag extraction from malware\_extracted folder

MALWARE\_DIR=~/malware\_extracted

FLAG\_OUTPUT=~/flag\_hits.txt

PATTERN='(?i)(flag\{[A-Za-z0-9\_]+\})'

# Remove previous flag file if exists

[ -f $FLAG\_OUTPUT ] && rm $FLAG\_OUTPUT

# Extract flags from all files in malware\_extracted

for FILE in $(find $MALWARE\_DIR -type f); do

CONTENT=$(cat "$FILE" 2>/dev/null)

if [[ $CONTENT =~ $PATTERN ]]; then

echo "${BASH\_REMATCH[0]}" >> $FLAG\_OUTPUT

fi

done

# List extracted flags

if [ -f $FLAG\_OUTPUT ]; then

echo "Flags recovered:"

cat $FLAG\_OUTPUT

else

echo "No flags found"

fi

# Zip all outputs for submission

zip -r ~/Desktop/submission\_project\_final.zip ~/pentest\_outputs ~/flag\_hits.txt ~/malware\_extracted VAPT\_Report\_Sanjana.txt 2>/dev/null || true

**Commands on Windows VM: Powershell**

# PowerShell Source Code for Vulnerability Assessment & Penetration Testing Project

# Author: Sanjana S

# Date: 23 Oct 2025

# Set destination folder for extracted malware and flag output

$Dest = "C:\Users\Administrator\Desktop\Malware\_extracted"

$Out = "C:\Users\Administrator\Desktop\flag\_hits.txt"

# Regex pattern to find flags

$Pattern = '(?i)(flag\{[A-Za-z0-9\_]+\})'

# Remove previous flag file if exists

if (Test-Path $Out) {

Remove-Item $Out -Force

}

# Extract flags from all files in malware\_extracted

Get-ChildItem $Dest -Recurse -File | ForEach-Object {

$Content = Get-Content $\_.FullName -Encoding ASCII -ErrorAction SilentlyContinue

$Matches = [regex]::Matches($Content, $Pattern)

if ($Matches.Count -gt 0) {

foreach ($m in $Matches) {

$m.Value | Out-File -Append -FilePath $Out

}

} else {

"flag not found in $($\_.FullName)" | Out-File -Append -FilePath $Out

}

}

# Process individual ZIP files (example: Lesson3)

$File = "C:\Users\Administrator\Desktop\Malware\_extracted\Lesson3-Multiple AV scanning\_malware.zip"

if (Test-Path $File) {

Add-Type -AssemblyName System.IO.Compression.FileSystem

[System.IO.Compression.ZipFile]::ExtractToDirectory($File, $Dest)

}

# List all flag files found

if (Test-Path $Out) {

Get-Content $Out

} else {

Write-Output "No flags extracted"

}

# Example HTTP server to share outputs

Set-Location "C:\Users\Administrator"

python -m http.server 8001

# Example of checking banner information manually

# (Assuming nmap or netcat equivalents in Windows if installed)

# Replace IP addresses and ports as per your lab setup

$IP = "10.0.2.2"

$Ports = @(22, 2289, 35768, 51078)

foreach ($Port in $Ports) {

try {

$tcpClient = New-Object System.Net.Sockets.TcpClient

$tcpClient.Connect($IP, $Port)

$stream = $tcpClient.GetStream()

$reader = New-Object System.IO.StreamReader($stream)

$banner = $reader.ReadLine()

$banner | Out-File -Append -FilePath $Out

$tcpClient.Close()

} catch {

Write-Output "Port $Port closed or unreachable" | Out-File -Append -FilePath $Out

}

}