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
    }
}
```