

Lab 9: Malware Persistence, Elevation of Privilege and Propagation Techniques

ITSC 303: Malware Analysis

Table of Contents

[Lab Outcomes 2](#_Toc492373015)

[Background Reading 2](#_Toc492373016)

[Introduction 2](#_Toc492373017)

[1.0 Persistence 3](#_Toc492373018)

[1.1 Persistence Dynamic Analysis 3](#_Toc492373019)

[1.2 Debugging Persistence Functions 6](#_Toc492373020)

[2.0 Propagation 7](#_Toc492373021)

[2.1 Ramnit HTML File Infection Propagation 7](#_Toc492373022)

[2.2 Ramnit Executable File Infection Propagation 9](#_Toc492373023)

[3.0 Privilege Escalation 9](#_Toc492373024)

[3.1 Non-Privileged PE Bind Shell 10](#_Toc492373025)

[3.2 Privilege Escalation Using AlwaysInstallElevated 12](#_Toc492373026)

[4.0 Sign-Off: Lab 9 – Malware Persistence, Elevation of Privilege and   
Propagation Techniques 14](#_Toc492373027)

[References 15](#_Toc492373028)

Malware Analysis

Lab 9: Malware Persistence, Elevation of Privileges and Propagation Techniques

Lab Outcomes

This lab will focus on the following outcomes:

* Analyze registry persistence mechanisms.
* Analyze privilege elevation mechanisms.
* Analyze propagation techniques.

Background Reading

* [Malware Persistence without the Windows Registry](https://www.fireeye.com/blog/threat-research/2010/07/malware-persistence-windows-registry.html) (www.fireeye.com/blog/threat-research/2010/07/malware-persistence-windows-registry.html)
* [Windows Registry Persistence, Part 2: The Run Keys and Search-Order](https://blog.cylance.com/windows-registry-persistence-part-2-the-run-keys-and-search-order) (https://blog.cylance.com/windows-registry-persistence-part-2-the-run-keys-and-search-order)
* [PowerSploit/Privesc/PowerUp.ps1](https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1) (https://github.com/PowerShellMafia/PowerSploit/blob  
  /master/Privesc/PowerUp.ps1)
* [Defeating Windows User Account Control](https://github.com/hfiref0x/UACME) (https://github.com/hfiref0x/UACME)

Introduction

Persistence, elevation of privilege and propagation are three main components of modern malware variants. Persistence involves using operating system functionality to maintain execution across reboots. Elevation of privilege involves changing access rights from a lower-privileged context (e.g., a normal user account) to a higher-privileged context (e.g., an administrator account) to gain access to as much of the system as possible. Propagation (e.g., self-replication) is the defining feature of viruses that differentiates them from benign computer programs.

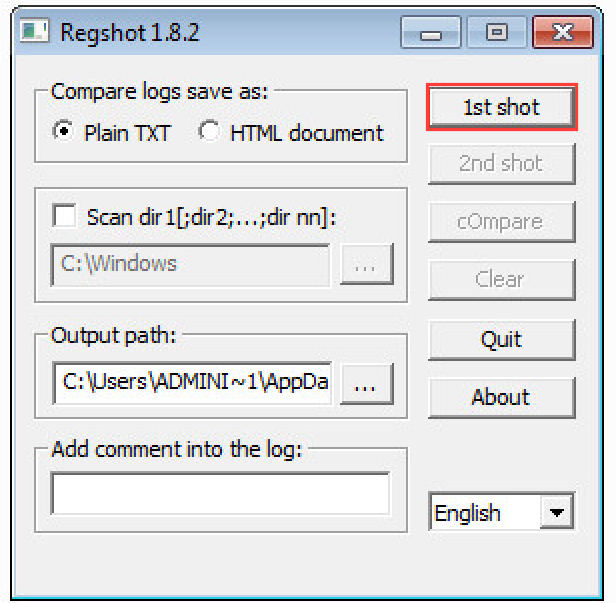
In this lab you will walk through a Ramnit virus sample that uses a service and the registry for persistence and infects files as a propagation vector. You will also use Metasploit’s Meterpreter tool to demonstrate a privilege escalation technique using MSI files.

1. Persistence

## Persistence Dynamic Analysis

There are a number of analysis tools that you can use to check registry persistence mechanisms. One of these tools is Regshot, which can be used to take two separate snapshots of the registry and can compare them to one another.

1. Open **Regshot** and click **1st shot**, as shown in Figure 1.



**Figure 1: Regshot 1st Shot**

Source: Regshot software, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

Regshot takes a snapshot of the current registry state for later comparison.

1. If you haven’t done so already, take a clean snapshot of your lab environment virtual machine.
2. Decompress the **ramnit.zip** file with the password **infected** and then infect the machine.
3. In Regshot, click **2nd shot** and then click **cOmpare**.

Regshot performs a diff of the two registry captures. Once the comparison is finished, Notepad opens and displays the results.

When new services are added using **CreateService**, or using the command line tool **sc**, registry keys are written to the registry key: HKLM\SYSTEM\ControlSet[0-9]{3}\Services. This is mirrored in HKLM\SYSTEM\CurrentControlSet\Services.

In the results, notice that there have been two additions in the *Keys added* section compared to the first registry shot taken.

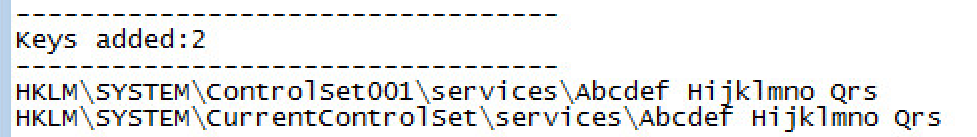


Figure 2: Registry Shot Comparison: Results Keys Added

Used with permission from Microsoft.

You’ll also notice two additions in the *Values modified* section compared to the first registry shot.

content/3-service-paths-added-after-compare.png

Figure 3: Registry Shot Comparison: ImagePath and Display Name

Used with permission from Microsoft.

This is a malicious service created for persistence by Ramnit. The ImagePath attribute provides the path to the service binary to be executed, and the DisplayName attribute is the service’s name.

Two additional attributes include Type and Start.

* The Type value is the sum of 0x10, representing SERVICE\_WIN32\_OWN\_PROCESS, and which designates the process as operating within its own process and not within svchost.exe.
* 0x100 represents SERVICE\_INTERACTIVE\_PROCESS, which designates the process as able to interact with the desktop.
* The Start value of 0x000000002 represents SERVICE\_AUTO\_START and designates the process to run at startup:

/Users/joshuareynolds/work/SAIT/CourseContent/Unit5/Week9/Lab/content/new2/3.3-service-paths-added-after-compare.png

Figure 4: Registry Shot Comparison: Type and Start

Used with permission from Microsoft.

1. Open Sysinternal’s Autoruns tool and select the **Logon** tab.

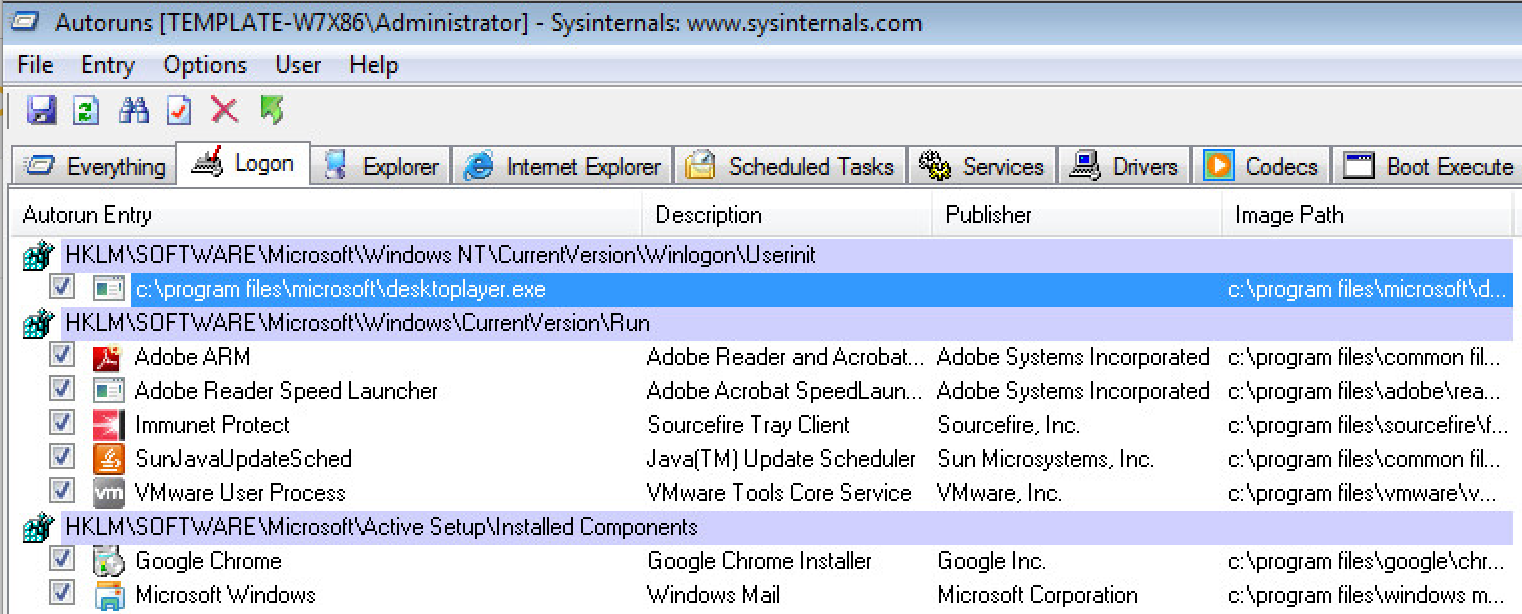


Figure 5: Autoruns Logon Tab Ramnit Userinit Persistence

Used with permission from Microsoft.

This tab provides a number of registry key entries that allow for persistence of executables at logon time. The entry under HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ Winlogon\Userinit points to c:\program files\microsoft\ desktoplayer.exe. This highlights how Autoruns provides another way to view persistence points of malware.

1. What is the specific purpose of this registry key?

**Other programs can be started from this key by appending them and separating them with a comma.**

**Instructor signoff:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Debugging Persistence Functions

Persistence techniques are not always as obvious as the one above. Persistence generation code is also often sought after by reverse engineers for writing broad indicators of compromise (IOC). It is useful to use dynamic analysis within a debugger to identify key functions for establishing persistence.

1. Revert the virtual machine to a clean state.
2. Open the provided Ramnit binary in OllyDbg.
3. Press CTRL+G and type **GetProcAddress** in the *Enter expression to follow* box. This is available because Kernel32.dll was loaded by the Windows PE loader.

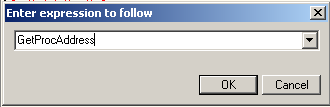


Figure 6: GetProcAddress Expression OllyDbg

Source: OllyDbg software, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

1. Set a software breakpoint on the function’s entry point.

A service is being created, but how the malware is doing this is currently unknown. The CreateProcess Windows API function is often a vector for this, which resides in Advapi32.dll. It is not possible to set a breakpoint on this function right now because the DLL is not loaded.

1. To break on this function, observe breakpoint hits to GetProcAddress until functions from within this DLL are dynamically loaded by Ramnit.
2. Press F9 to run the binary. Continue to hit this breakpoint until the ProcNameOrOrdinal parameter points to OpenSCManagerA.

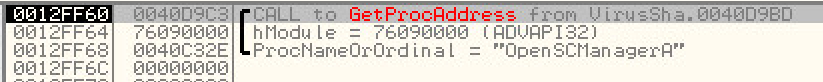


Figure 7: GetProcAddress Hit for OpenSCManagerA

Source: OllyDbg software, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

The hModule parameter points to ADVAPI32. For this to be possible, the DLL has to be in memory.

1. Now that the library is loaded, navigate to the CreateServiceA function, and set a software breakpoint at its entry point, as you did with GetProcAddress.
2. Continue execution using F9.

The breakpoint is hit, as shown in Figure 8.

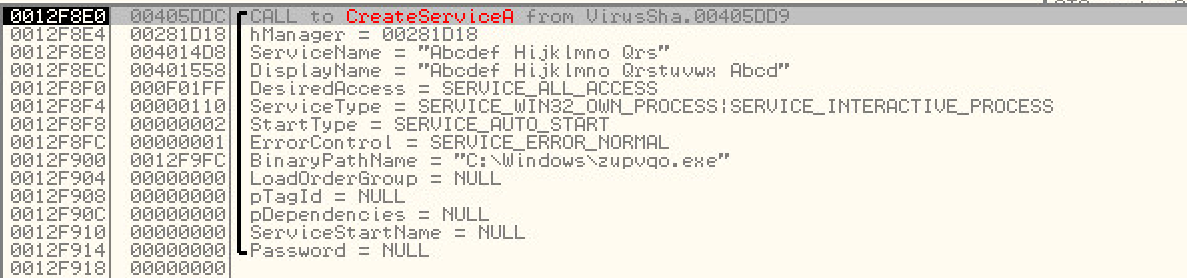


Figure 8: CreateServiceA Software Breakpoint Hit

Source: OllyDbg software, 2016. Reproduced and used in accordance with the fair dealing provisions in section 29 of the Canadian Copyright Act for the purposes of education, research or private study. Further distribution may infringe copyright.

The service attributes observed earlier are supplied as parameters to this function.

**Instructor signoff:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Propagation

Propagation mechanisms take many forms, one of the most prevalent being file infection. This involves malware adding its own code to files on the system, which used to be benign, in order to self-propagate. This section leads you through an investigation of Ramnit’s portable executable and HTML file infection capabilities.

## Ramnit HTML File Infection Propagation

To identify infected files, use the write timestamps.

1. Once again, revert your analysis environment to a clean state.
2. Infect the machine with Ramnit.
3. From PowerShell, navigate to your system’s root directory and run the following command (replace the date with yesterday’s date):

get-childitem -R | ? {$\_.lastwritetime -gt '10/29/16'}

1. Open one of the resulting .html files. It should contain infected HTML code, as shown in Figure 9.

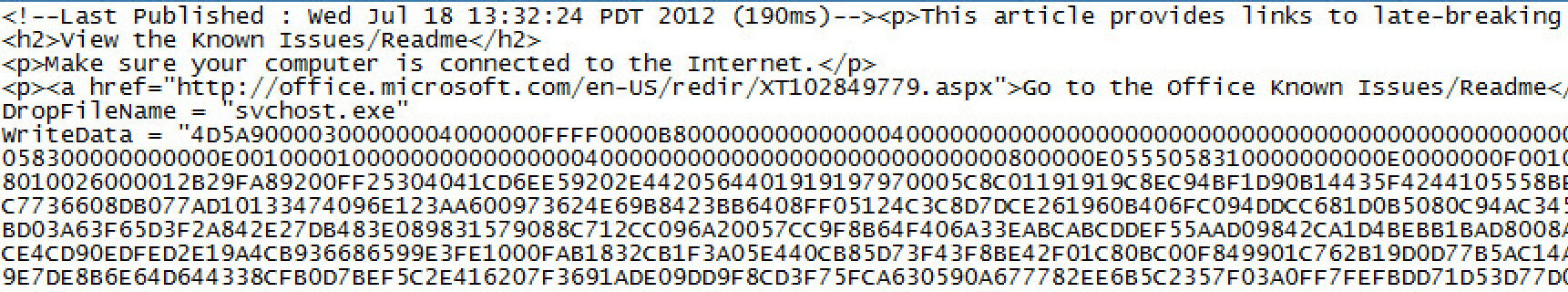


Figure 9: Ramnit-Infected HTML File

Used with permission from Microsoft.

**Questions**

1. What file is being written by the propagation code? What format is it currently stored in the infection code? How is this format being converted to binary code?

**It is writing a program called “svchost.exe”.**

**It is stored as raw opcode data**

**FileObj.Write Chr(CLng(“&H” & Mid(WriteData,I,2)))**

**That line is what converts it from hex to binary**

1. What scripting language is the propagation code written in? What makes this evident in the HTML?

**<SCRIPT Language=VBScript><!—**

**This line seems to indicate that the scripting language is VB Script**

1. How is this code executing the dropped file?

**It is spawning a shell**

**Instructor signoff:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Ramnit Executable File Infection Propagation

1. Using the output from the PowerShell command mentioned above, find an executable that has been infected by Ramnit.
2. Open the executable in Hiew and make note of section names, their addresses and the executable’s entry point.
3. Revert the system to a clean state and open the same executable in Hiew.

**Questions**

1. Are there any additional sections in comparison? If so, what are the RVA and Offset of the new section (or sections)?

**As we do not have access to Hiew, most of this information cannot be found and as such I am skipping this section.**

1. Has the binary’s entry point been modified? If so, where does the new EP point to?

**Instructor signoff:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Privilege Escalation

This section demonstrates privilege escalation techniques using binaries from the Metasploit Framework. Metasploit can produce various forms of malicious portable code. In this instance, you’ll misconfigure the Windows instance on purpose to achieve privilege escalation using an MSI bind shell.

## Non-Privileged PE Bind Shell

1. Revert the analysis machine to a clean state.
2. Copy **bind-shell-tcp.exe** to the machine and execute it.

This shell binds to port 4444. You are prompted to allow connections to this application.

1. Click **Allow access**.

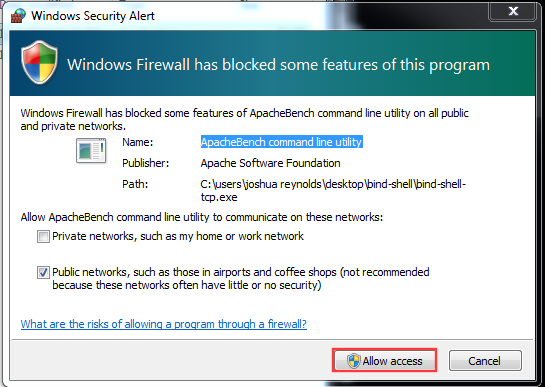


Figure 10: Allow Firewall Access TCP Bind Shell

Used with permission from Microsoft.

1. Connect to this bind shell and launch a command prompt using:

telnet [IP of analysis machine]

This results in:

$ telnet 192.168.118.128 4444

Trying 192.168.118.128...

Connected to 192.168.118.128.

Escape character is '^]'.

Microsoft Windows [Version 6.1.7601]

Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\test-user\Desktop\bind-shell>

1. Check your current permissions using whoami.

C:\Users\test-user\Desktop\bind-shell>whoami

whoami

win-fcit718ht1r\test-user

This is a non-privileged context. Verify this by attempting to write to your root directory.

C:\Users\test-user\Desktop\bind-shell>echo "test" > C:\test

echo "test" > C:\test

Access is denied.

## Privilege Escalation Using AlwaysInstallElevated

We’re going to be editing registry settings ourselves, however, it is common for misconfigurations to be present in Windows environments, which may be exploited by malware variants to gain elevated privileges.

1. Open **regedit**, navigate to **HKEY\_LOCAL\_MACHINE\Software\Policies\ Microsoft\Windows\** and create the **Installer** registry key.

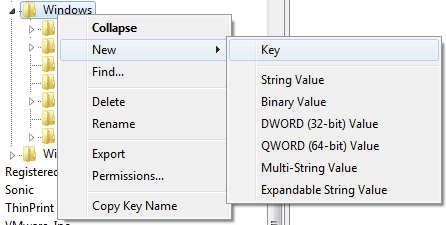


Figure 11: Create Install Registry Key

Used with permission from Microsoft.

1. Following the same path as above, select **New > DWORD 32-bit Value** and create a new value called **AlwaysInstallElevated**. Set its value to **1**.

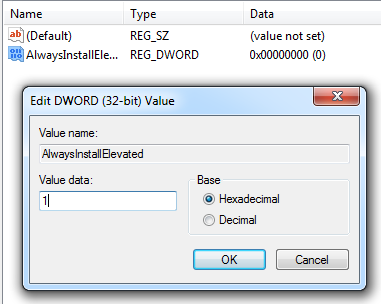


Figure 12: Set AlwaysInstallElevated to 1

Used with permission from Microsoft.

1. Set the same value under **HKEY\_CURRENT\_USER\Software\Policies\Microsoft\Windows\**.
2. Execute **bind-shell-tcp.msi** and provide firewall permissions again.
3. Once the MSI file executes, telnet to port 5555 and execute **whoami** to verify elevated privileges.

C:\Users\test-user\Desktop\bind-shell>whoami

nt authority\system

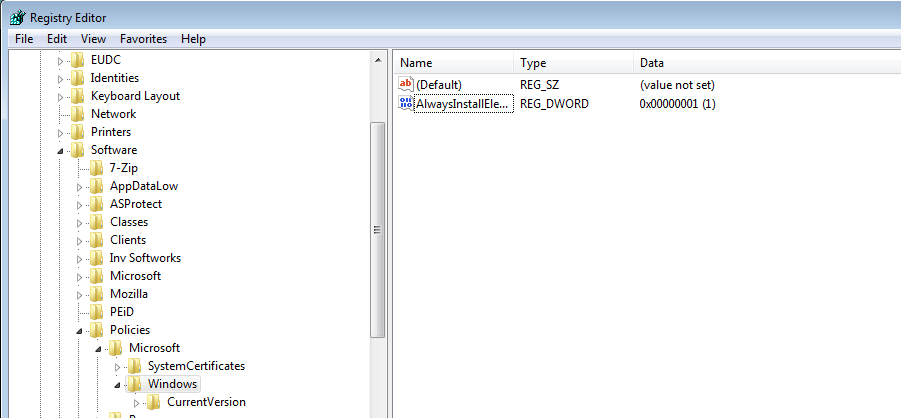
Due to the misconfiguration you are now **nt authority\system**, which is the highest Windows access privilege.

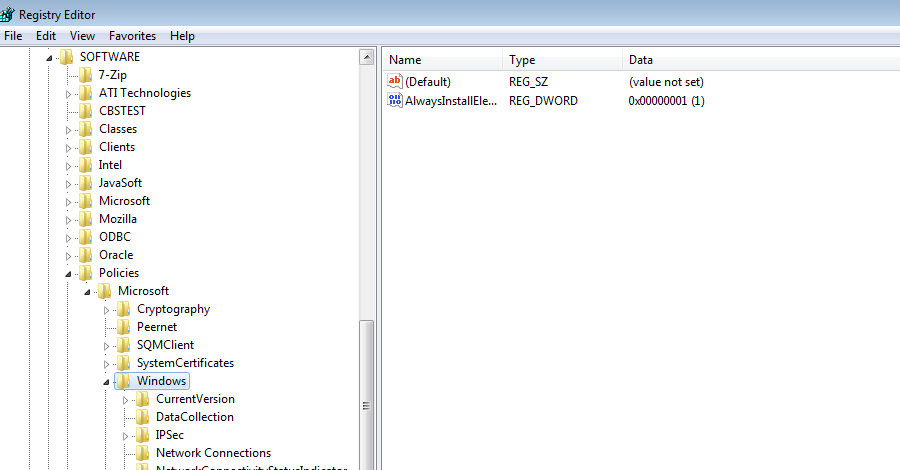
1. Verify your escalated privileges by writing to the same test file:

C:\Users\test-user\Desktop\bind-shell>echo "test" > C:\test

echo "test" > C:\test

I have completed all the steps above, however I will not be connecting this VM to the internet and accessing it remotely as I do not want any of this malware to spread onto any of my other systems. Here are screen shots of the edited registry.





**Instructor signoff:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Sign-Off: Lab 9 – Malware Persistence, Elevation of Privilege and Propagation Techniques

Detach this page and submit it to your instructor.

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| **Item** | **Initial** |
| Persistence Dynamic Analysis |  |
| Debugging Persistence Functions |  |
| Ramnit HTML File Infection Propagation |  |
| Ramnit Executable File Infection Propagation |  |
| Privilege Escalation Using AlwaysInstallElevated |  |

# References

Regshot (Version 1.9.0.7z) [Computer software]. Retrieved from https://sourceforge.net/projects/regshot/.

Yuschuk, O. (2014). OllyDbg (Version 2.01) [Computer software]. Retrieved from http://www.ollydbg.de/.