Chapter 6
Enumeration

THE FOLLOWING CEH EXAM TOPICS ARE COVERED IN THIS CHAPTER:

- ✓ **Technical assessment methods**
- ✓ **Network security**
- ✓ **Vulnerabilities**
- ✓ **Application/file server**

Port scanning is ultimately about identifying applications that are installed on systems within the target network. Once we have identified applications, though, we will want to dig deeper to see what additional information we can extract. This may include user information or details about shares that may be available on the network. Of course, there are other activities we can perform when we start working on enumeration. This information gathering will be beneficial when we start moving to the next stages.

Enumeration is about determining what services are running and then extracting information from those services. The first thing you need to do is identify services that are available on your target systems. Each service may have a lot of information that can be obtained. External-facing services may have authentication requirements, which means there are users. As an example, users may have to be authenticated and authorized to view some sections on a web server. You may be able to get the web server to give you an indication what usernames are configured on the server, which would be an example of enumeration.

Because we are working on enumeration, we are going to take a close look at a few protocols as well as the tools you would use with those protocols. For a start, there is the Server Message Block (SMB) protocol. This is used on Windows systems for file and resource sharing as well as some remote

management. This is definitely a case where users would have to authenticate against the service, so we can spend some time trying to find users on Windows servers. Additionally, you may be able to identify security policy information associated with the Windows domain. Certainly, you should be able to identify file shares where there are some.

Other protocols you may not think about when it comes to enumeration are the Simple Mail Transfer Protocol (SMTP) and the Simple Network Management Protocol (SNMP). It's common for users to have to authenticate and be authorized before sending email through an SMTP server, particularly if they are sending from outside the network where the mail server is. If you use a traditional mail client to connect with Gmail or Office 365, for example, you are familiar with having to provide your username and password for your SMTP server. Your client may automatically fill that information in for you, but it's there if you go looking at settings.

SNMP can provide a lot of information about systems. If you can get access to an SNMP system, you should be able to walk the management information base (MIB) to extract details from your target system. There are tools that will perform this walk for you, retrieving the information and presenting it to you.

The MITRE ATT&CK framework categorizes enumeration under the Reconnaissance phase. It currently identifies Gather Victim Host Information and Gather Victim Identity Information as two techniques that would broadly fit into the enumeration category.

Services are necessary since they are how work gets done. You can't, for example, send email without there being an email server somewhere to receive the message. This is what attackers take advantage of. Services that may be vulnerable or used for exploitation are a guarantee there is an application that is accessible remotely. This does not mean, however, that services can't be protected from enumeration. These countermeasures are ways to help be sure legitimate users are accessing or probing the service for information.

By the time we are done with this chapter, you should have a solid understanding of what enumeration is as well as what tools you can use to enumerate different resources on systems. Many of these tools will be

Linux-based and run from the command line, but there are some Windows tools we'll look at as well.

Service Enumeration

When you are scanning systems, `nmap` is always your friend. The same is true when it comes to service enumeration. This means you are identifying the service running on the target system. A quick way to do that is to use the version scan built into `nmap`. In the following code listing, you can see a portion of output from a version scan run by `nmap` on hosts on my network. A version scan is performed by using `-sV` as the parameter sent to `nmap`. It shows not just open ports but also where it can find them and specifics about the services and versions that are running on the hosts that were found responding on the network. It does this by looking at any application banners to extract details about the service name and version.

nmap Version Scan

```
 $ sudo nmap -sV 192.168.1.15
 Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-12 13:44
EST

 PORT    STATE  SERVICE VERSION
 22/tcp  open   ssh     OpenSSH 7.7p1 Debian 3 (protocol 2.0)
 25/tcp  closed smtp
 80/tcp  open   http    Greenbone Security Assistant
 443/tcp closed https
 MAC Address: 0E:76:03:B8:2A:BA (Unknown)
 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

 Nmap scan report for desktop-3rgc5h2.lan (192.168.86.60)
 Host is up (0.025s latency).

 PORT    STATE    SERVICE VERSION
 22/tcp  filtered ssh
 25/tcp  filtered smtp
 80/tcp  filtered http
 443/tcp filtered https
 MAC Address: C4:9D:ED:AB:DD:7A (Microsoft)

 Nmap scan report for milobloom.lan (192.168.86.61)
 Host is up (0.94s latency).
```

```
PORT    STATE  SERVICE VERSION
22/tcp  open   ssh     OpenSSH 7.6 (protocol 2.0)
25/tcp  closed smtp
80/tcp  closed http
443/tcp closed https
MAC Address: B8:09:8A:C7:13:8F (Apple)
```

As you can see, not all services provide details about what they are. We can identify the service but not the application or the version in most cases. One thing we can see in the previous listing is a handful of systems that are running Secure Shell (SSH). Not all of the SSH servers provided versions or even protocols. Fortunately, we can make use of `nmap` again for more details about SSH. `nmap` has scripting capabilities, and there are a lot of scripts that will enumerate services for more details. One of these scripts will enumerate algorithms that are supported by the SSH server. SSH encrypts data between the client and the server, but the cipher suites used may vary between connections, since clients can support different key strengths and algorithms. Here you can see the use of the script used to enumerate the algorithms across SSH servers, `ssl-enum-ciphers.nse`.

SSH2 Algorithm Enumeration

```
 $ sudo nmap --script=ssl-enum-ciphers 192.168.1.15
 Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-12 13:46
EST
 Nmap scan report for 192.168.1.15
 Host is up (0.073s latency).
 Not shown: 997 closed tcp ports (reset)

 PORT   STATE SERVICE
 22/tcp open  ssh
 | ssh2-enum-algos:
 |   kex_algorithms: (10)
 |       curve25519-sha256
 |       curve25519-sha256@libssh.org
 |       ecdh-sha2-nistp256
 |       ecdh-sha2-nistp384
 |       ecdh-sha2-nistp521
 |       diffie-hellman-group-exchange-sha256
 |       diffie-hellman-group16-sha512
 |       diffie-hellman-group18-sha512
 |       diffie-hellman-group14-sha256
 |       diffie-hellman-group14-sha1
```

```
|   server_host_key_algorithms: (5)
|       ssh-rsa
|       rsa-sha2-512
|       rsa-sha2-256
|       ecdsa-sha2-nistp256
|       ssh-ed25519
|   encryption_algorithms: (6)
|       chacha20-poly1305@openssh.com
|       aes128-ctr
|       aes192-ctr
|       aes256-ctr
|       aes128-gcm@openssh.com
|       aes256-gcm@openssh.com
|   mac_algorithms: (10)
|       umac-64-etm@openssh.com
|       umac-128-etm@openssh.com
|       hmac-sha2-256-etm@openssh.com
|       hmac-sha2-512-etm@openssh.com
|       hmac-sha1-etm@openssh.com
|       umac-64@openssh.com
|       umac-128@openssh.com
|       hmac-sha2-256
|       hmac-sha2-512
|       hmac-sha1
|   compression_algorithms: (2)
|       none
|_      zlib@openssh.com
MAC Address: 0E:76:03:B8:2A:BA (Unknown)
```

You will see a collection of algorithm types in the output. The first set of algorithms is for key exchange. One of them is the Diffie-Hellman algorithm, named for Whitfield Diffie and Martin Hellman, who were the first to publish an algorithm for key exchange. The key exchange algorithm is important because the key is essential for encryption, and it is generated at the time a connection is made. You can also see the encryption algorithms listed. Most of these are the Advanced Encryption Standard (AES), though you'll notice one is named ChaCha20. This is a stream cipher like AES that can allow programs to use encryption without the need for an open source encryption library. Finally, there is the message authentication code, used to ensure that the message wasn't tampered with or corrupted.

Diffie and Hellman were the first to publish a key exchange algorithm, but they were not the first to develop one. Government intelligence agencies had already come up with key exchange algorithms separately from Diffie and Hellman. The difference was that the agency employees were unable to publish because their work couldn't be disclosed outside the agency.

Certainly `nmap` can provide you with a good start on getting a list of services and version numbers, but it's not always enough. There is much more that can be acquired about different services. This is an area that `nmap` can help with through the use of scripts that are installed with `nmap`. These scripts can be used to extract a lot of information, like the algorithms in SSH, that are otherwise difficult to attain. SSH may provide encryption services, but not all of the encryption algorithms are free from vulnerabilities. This is why it can be useful to know everything about services. You never know where you may run across a vulnerability.

Countermeasures

Each service is going to have different configuration settings that will prevent attackers from enumeration, but at a high level, the following countermeasures should be considered for every service:

Firewalls Both network and host-based firewalls can be used to protect services by preventing systems that should not be communicating to send requests.

Authentication Strong authentication can help to protect some services, since authentication may happen before requests can be made to the service.

Reduce Information Provided Some services will allow you to configure how much information is provided in the banner—the information the service greets the user with when the application receives a connection request.

Remote Procedure Calls

A remote procedure call (RPC) is a service that allows remote systems to consume procedures external to the application calling them. A program on system A can call a function or procedure on another system across the network. It does this using the RPC protocol. As far as the program on the local computer calling the remote procedure is concerned, it's a local procedure existing in the same process space as the rest of the code. The program calls this procedure, gets the information, and proceeds on its merry way. RPCs provide a way for two processes to communicate with one another. *Remote* commonly means a remote server, but two local processes could also use RPCs to communicate with one another.

SunRPC

The idea of interprocess communication has been around for decades. There have been several implementations of request-response protocols over the decades. Java's remote method invocation (RMI) is a recent example of this. Before that, there was the Common Object Request Broker Architecture (CORBA), which was independent of language implementation. Sometimes, with RPCs, you need what is essentially a directory service to indicate the dynamic ports on which different services are running.

A common implementation of remote procedure calls is the program `portmap`, also known as `rpcbind`. This program is used to provide information about programs that have been registered with the portmapper service, providing these remote procedures. The portmapper assigns a port for the service to listen on and, when queried, can provide that information back. Common examples of services that use `rpcbind`/`portmap` are file sharing servers like Network File Server (NFS).

The package that provides the `portmap` or `rpcbind` service may also provide utilities that can also communicate using RPC. This is done over port 111. To identify programs and associated ports on a remote system, you can use the program `rpcinfo`. You can see an example of the use of `rpcinfo` shown here. The command used, `rpcinfo -p`, has the `rpcinfo` probe the host provided. In this case, the host is an IP address rather than a hostname.

rpcinfo List

```
kilroy@bobbie $ rpcinfo -p 192.168.86.52
    program vers proto   port  service
     100000    4   tcp    111  portmapper
     100000    3   tcp    111  portmapper
     100000    2   tcp    111  portmapper
     100000    4   udp    111  portmapper
     100000    3   udp    111  portmapper
     100000    2   udp    111  portmapper
     100005    1   udp  43939  mountd
     100005    1   tcp  58801  mountd
     100005    2   udp  46384  mountd
     100005    2   tcp  50405  mountd
     100005    3   udp  49030  mountd
     100005    3   tcp  50553  mountd
     100003    3   tcp   2049  nfs
     100003    4   tcp   2049  nfs
     100227    3   tcp   2049  nfs_acl
     100003    3   udp   2049  nfs
     100227    3   udp   2049  nfs_acl
     100021    1   udp  34578  nlockmgr
     100021    3   udp  34578  nlockmgr
     100021    4   udp  34578  nlockmgr
     100021    1   tcp  39297  nlockmgr
     100021    3   tcp  39297  nlockmgr
     100021    4   tcp  39297  nlockmgr
```

The programs shown earlier that have remote procedures registered with `rpcbind` are associated with the NFS file sharing server. The program portmapper is the primary service that is queried for additional data, but the others, like `mountd`, `nfs`, and `nlockmanager`, are all needed for NFS. NFS was developed by Sun Microsystems. The portmapper is an implementation of RPC that was also associated with Sun. You may sometimes see it referred to as sunrpc. This is the case with a scanner in Metasploit that can also be used to identify the ports allocated to programs using the portmapper.

Metasploit sunrpc Scanner

```
 msf> use auxiliary/scanner/misc/sunrpc_portmapper

 msf auxiliary(scanner/misc/sunrpc_portmapper)> set RHOSTS
192.168.86.52
 RHOSTS => 192.168.86.52
 msf auxiliary(scanner/misc/sunrpc_portmapper)> run
```

```
[+] 192.168.86.52:111 - SunRPC Programs for 192.168.86.52
================================

Name      Number  Version  Port   Protocol
----      ------  -------  ----   --------
mountd    100005  1        43939  udp
mountd    100005  1        58801  tcp
mountd    100005  2        46384  udp
mountd    100005  2        50405  tcp
mountd    100005  3        49030  udp
mountd    100005  3        50553  tcp
nfs       100003  3        2049   tcp
nfs       100003  4        2049   tcp
nfs       100003  3        2049   udp
nfs_acl   100227  3        2049   tcp
nfs_acl   100227  3        2049   udp
nlockmgr  100021  1        34578  udp
nlockmgr  100021  3        34578  udp
nlockmgr  100021  4        34578  udp
nlockmgr  100021  1        39297  tcp
nlockmgr  100021  3        39297  tcp
nlockmgr  100021  4        39297  tcp
rpcbind   100000  4        111    tcp
rpcbind   100000  3        111    tcp
rpcbind   100000  2        111    tcp
rpcbind   100000  4        111    udp
rpcbind   100000  3        111    udp
rpcbind   100000  2        111    udp

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

As we are scanning the same host, it's not unexpected that we'd get the same results using the Metasploit module as we got from `rpcinfo`. Since all that is happening is querying the portmapper process, you can use whatever tool makes you the most comfortable. There are a couple of advantages to using these tools over `nmap`. The first is that you'll get the process name by checking with portmapper. The second is that you won't get all of the ports using `nmap` unless you specifically indicate that you want to scan all ports. The range of ports handed out by `portmap` isn't in the list of well-known ports that `nmap` scans.

Remote Method Invocation

Java programs are very popular, especially when it comes to web applications. Java provides a lot of assistance to programmers, especially when it comes to libraries and interfaces that can be implemented with your own functionality. Java includes its own capability for remote procedure calls, though in Java it's called *remote method invocation*. To have a program that uses RMI, the system needs a version of the portmapper for Java called the `rmiregistry`. The program using RMI registers itself with the `rmiregistry` program. This means that anyone can check with the `rmiregistry` to see what services are offered. The `rmiregistry` program will respond in a similar way to what we saw when we checked with the portmapper.

It's said that RMI is the object-oriented version of RPC. This means objects get passed between the server and the client. The client implements a stub through an interface. An interface is an object-oriented term indicating a definition of a class. The stub communicates with a skeleton on the server. When a programmer is creating a program that uses RMI, they use an RMI compiler (the program `rmic`). The programs we use to connect to an RMI registry to enumerate services that are registered don't need to know the specific interfaces needed to pass objects between the skeleton and the stub because the only thing the enumeration is doing is identifying the skeletons or services on the remote system. We'll start with Metasploit to run a scan on a system that has RMI. You can see an example of using Metasploit for RMI enumeration here.

Running RMI Scanner in Metasploit

```
msf> use auxiliary/gather/java_rmi_registry
msf auxiliary(gather/java_rmi_registry)> show options

Module options (auxiliary/gather/java_rmi_registry):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  1099             yes       The target port (TCP)

msf auxiliary(gather/java_rmi_registry)> set RHOST
192.168.86.62
RHOST => 192.168.86.62
msf auxiliary(gather/java_rmi_registry)> run
```

```
 [*] 192.168.86.62:1099 - Sending RMI Header…
 [*] 192.168.86.62:1099 - Listing names in the Registry…
 [+] 192.168.86.62:1099 - 1 names found in the Registry
 [+] 192.168.86.62:1099 - Name HelloServer (HelloImpl_Stub)
found on 127.0.1.1:38371
 [*] Auxiliary module execution completed
```

The options required to run the `java_rmi_registry` module are simple. The remote port defaults to 1099, which is the port `rmiregistry` listens on. If you think there is another port listening, you can set the RPORT variable. The one that is essential, though, is RHOST. We set this variable to the IP address of the system where there is a simple Java program that implements an RMI server. You can see the result of the scan. According to the RMI registry, there is a server named `HelloServer`. It even tells us that that the stub is named `HelloImpl`. Even though it's a server, the registry calls it a stub because the difference between a stub and a skeleton is the end of the conversation that's happening. The `rmic` generates stubs for both servers and clients. It's just that referring to the server end as a skeleton differentiates between the two.

Metasploit is not the only way you can scan for RMI services. If you look around a little, you can find additional programs. One of these is called Barmie, stylized as BaRMIe to highlight the RMI in the name. You can grab the source code as well as a precompiled implementation of BaRMIe through GitHub. Running the program is straightforward. Since it's a Java program, you have to run the intermediate code file through the Java program used to create a Java virtual machine. Since it's stored as a Java archive (JAR), you have to tell Java that it is going to be running from one of those. The program does need to know what host it's scanning, so you pass in the IP address of your target. You can see a run of BaRMIe next.

The program `javac` is used to compile Java source code to an intermediate code file. The program `java` is used to execute an intermediate code file.

Using BaRMIe

```
root@quiche:~# java -jar BaRMIe_v1.01.jar  192.168.86.62
```

```
  |</line><line xml:id="c06-line-0158"><![CDATA[ |

   |    |   |
                                                        v1.0
                      Java RMI enumeration tool.
                        Written by Nicky Bloor (@NickstaDB)

 Warning: BaRMIe was written to aid security professionals in
identifying the
          insecure use of RMI services on systems which the
user has prior
          permission to attack. BaRMIe must be used in
accordance with all
          relevant laws. Failure to do so could lead to your
prosecution.
          The developers assume no liability and are not
responsible for any
          misuse or damage caused by this program.

 Scanning 1 target(s) for objects exposed via an RMI registry…

 [-] An exception occurred during the PassThroughProxyThread
main loop.
     java.net.SocketException: Socket closed
 [-] An exception occurred during the
ReplyDataCapturingProxyThread main loop.
     java.net.SocketException: Socket closed
 RMI Registry at 192.168.86.62:1099
 Objects exposed: 1
 Object 1
   Name: HelloServer
   Endpoint: 127.0.1.1:38371
   [+] Object is bound to localhost, but appears to be exposed
remotely.
   Classes: 3
     Class 1
       Classname: java.rmi.server.RemoteStub
     Class 2
```

```
      Classname: java.rmi.server.RemoteObject
    Class 3
      Classname: HelloImpl_Stub

 1 potential attacks identified (+++ = more reliable)
 [---] Java RMI registry illegal bind deserialization

 0 deserialization gadgets found on leaked CLASSPATH
 [~] Gadgets may still be present despite CLASSPATH not being
leaked

 Successfully scanned 1 target(s) for objects exposed via RMI.
```

We see a couple of things from running this program. It's a little more verbose than Metasploit is. We get the name of the server, `HelloServer`. Just as with Metasploit, we see that the server is bound to the localhost on port 38371, which was dynamically allocated by the `rmiregistry`. We also see the inheritance tree from this. We can see references to the classes `java.rmi.server.RemoteStub`, `java.rmi.server.RemoteObject`, and `HelloImpl.Stub`. According to BaRMIe, the service is exposed remotely but is available only to localhost. This means we have information leakage. To attempt to identify vulnerabilities with that RMI server and potentially exploit those vulnerabilities, we need to gain access to the system.

In identifying the RMI registry and the additional services, we have also identified the existence of another piece of software on the target system. It may seem obvious now, but if you find an RMI registry and RMI services, you have found a system that at least has a Java runtime engine (JRE) on it, if not a Java development kit (JDK). What we don't know from the output here is the version of the JRE or JDK. However, there have been vulnerabilities in Java implementations over the last few years. Knowing there is at least a JRE on the system may have given you a lead to vulnerabilities.

Server Message Block

The most common implementation of remote procedure calls you will run across is the one used in Windows networks. The SMB protocol is complex when you consider all the different ways it can be used and all the different

ways it will operate. You may be most familiar with SMB as the protocol used to share files across a network. While this is definitely one use for SMB, it is not the only one, and even when you think about sharing files, there is a lot more than just transmitting file contents across a network.

SMB is an Application layer protocol that can operate over different protocols at lower layers. First, it can operate directly over TCP without any other Session layer protocols. If a system were running SMB directly over TCP, you would find TCP port 445 to be open. SMB can also operate over session protocols like NetBIOS, which is an application programming interface (API) developed by IBM to extend the input/output (I/O) capabilities away from the local system and onto the network. If you see UDP ports 137 and 138 open, you will know that you have found SMB running on top of NetBIOS. However, if you find TCP ports 137 and 139 open, you will have found SMB running on NetBIOS over TCP. Keep in mind that NetBIOS is used for name services in this case.

So, just what is SMB good for? SMB is used for communicating between Windows systems—file sharing, network management, system administration. This may mean managing naming of systems to be certain there aren't conflicts. Management like this requires that systems announce themselves to the local network. SMB also has to support authentication so systems aren't wide open to the entire network. This means SMB knows about users and groups. It also knows about shares, which are directories that are exposed to the network. Systems have to be able to provide the list of shares they are exporting to the network to systems that ask. This allows a user to get a list of shares and then access the ones they want, after authentication.

Authentication is not always necessary. SMB supports something called *null authentication*. What this means is there are some functions that don't require a username and password. A system can request information about another system on the network using null authentication, meaning no credentials were passed. This null authentication can allow us to gather a lot of information about the system.

We can use several different tools to enumerate information on Windows systems. Actually, it's not even just Windows systems, though the intent of implementing SMB on other systems is to interoperate with Windows

systems. Samba is a package that can be installed on Unix-like operating systems, providing SMB as well as a NetBIOS naming service. There are two separate processes that are used by Samba. One is smbd, which handles SMB, and there is also nmbd, which handles the naming aspects of interoperating with Windows systems. This means that even while we are looking to enumerate information from Windows systems, we can also scoop up Unix-like systems.

The first place to start is using built-in tools. Built-in tools are especially available on Windows systems, but there are Unix-like utilities as well. We will also look at a number of plugins available for `nmap` to gather information. Metasploit, not surprisingly, has several modules for scanning SMB systems. There are also some other utilities you can use, and we'll take a look at some of those.

Built-in Utilities

If you are on a Windows system, there are a number of utilities that you can make use of to gather information using SMB. Analogs exist for Linux as well. One thing to make note of with regard to the built-in utilities is that you need to be on the same broadcast domain to make use of them. NetBIOS was originally developed to be used in a local area network, rather than with the concept of wide area networks built in. As a result, some of the functions work because systems rely on broadcasting information to the network. The implication of this is that you need to have a presence on the local network before these utilities will work.

Gathering NetBIOS statistics can be accomplished by using the program `nbtstat`. This allows you to gather data about the local network. In the following example, you can see the use of `nbtstat` to acquire data about a remote system. Using `nbtstat -a` presents the name table for the hostname provided. If all we knew was the IP address of the remote system, we could use `nbtstat -A` instead. What you'll see is that we get different pieces of information. You get a list of names down the left side, followed by a code. The code indicates the context in which the name exists, followed by the status of each name.

nbtstat Output

```
C:\Users\kilroy> nbtstat -a billthecat

Local Area Connection:
Node IpAddress: [192.168.86.50] Scope Id: []

           NetBIOS Remote Machine Name Table

       Name               Type         Status
    ---------------------------------------------
    BILLTHECAT     <00>  UNIQUE      Registered
    BILLTHECAT     <20>  UNIQUE      Registered
    WORKGROUP      <00>  GROUP       Registered

    MAC Address = AC-87-A3-36-D6-AA
```

The codes shown are from NetBIOS, and you can look up the code to determine what the name is associated with. Systems that use SMB have a number of contexts in which they can exist. What you see here is a system that is both a workstation and a file server. This means that file sharing has been enabled on this system. Additionally, though, the system acts as a workstation or client on the network. It's important to distinguish the capabilities because then each system can know the types of questions that can be asked of each of the other systems. In technical terms, each set of functionalities has procedures associated with it. We can't call procedures that don't exist, so before procedures are called on the remote systems to initiate an action, we have to know what procedures are available. `nbtstat -a` essentially provides that information.

What we've seen so far simply asks for all the functions (names) associated with a hostname on the network. That's one individual system. If we want to see all the hostnames that are talking SMB/NetBIOS, we need to ask for something else. We can still use `nbtstat`, we just pass a different parameter in on the command line. We are looking for resolved names. This list can come from broadcast messages when there is no centralized database for name lookups—systems announce their names and their presence when they come online and then periodically after that. It can also come from the Windows Internet Name Server (WINS), which is a central repository of names of systems on an enterprise network. Windows servers will have

WINS functionality, so systems register with the WINS and all names can be resolved.

In the following code listing, you can see a list of names on the network. Since there is no Windows server and, as a result, no WINS on the network, these are all names that have been identified through broadcast messages. These systems are all macOS, but they are sharing files on the network using SMB. To do that, they need to behave like any other system that communicates using SMB.

Listing Resolved Names with nbtstat

```
C:\Users\kilroy
> nbtstat -r

        NetBIOS Names Resolution and Registration Statistics
        ----------------------------------------------------

        Resolved By Broadcast     = 47
        Resolved By Name Server   = 0

        Registered By Broadcast   = 8
        Registered By Name Server = 0

        NetBIOS Names Resolved By Broadcast
        -----------------------------------------------
                YAZPISTACHIO   <00>
                BILLTHECAT     <00>
                YAZPISTACHIO   <00>
                YAZPISTACHIO   <00>
                LOLAGRANOLA    <00>
                LOLAGRANOLA    <00>
                YAZPISTACHIO   <00>
                YAZPISTACHIO   <00>
```

There are other functions that `nbtstat` offers, but they are more related to functionality of the local system and less relevant for enumeration. While `nbtstat` has a lot of functionality, it is, as noted earlier, only on Windows systems. There are other tools you can use if you aren't running on Windows. If you have a Linux system and have the Samba package installed, which provides services that allow Linux to communicate using SMB, you can make use of the tool `nmblookup`. This can be used to do

lookups of names on the network. It can be used to query WINS as well as look up names where the systems are just broadcasting their information. For example, to get the details about the system `billthecat` as we did earlier, you would use `nmblookup -S -B billthecat`, as you can see here.

nmblookup for Enumeration

```
kilroy@savagewood$ nmblookup -S -B 192.168.86.255 billthecat
Can't load /etc/samba/smb.conf - run testparm to debug it
querying billthecat on 192.168.86.255
192.168.86.32 billthecat<00>
Looking up status of 192.168.86.32
     BILLTHECAT      <00> -          H <ACTIVE>
     BILLTHECAT      <20> -          H <ACTIVE>
     WORKGROUP       <00> - <GROUP> H <ACTIVE>

     MAC Address = AC-87-A3-36-D6-AA
```

Using `-B` tells `nmblookup` to use the broadcast address that is supplied, which is just the broadcast address on the local network. To use WINS, you could use `-R` to do a recursive lookup on the name. The flag `-S` tells `nmblookup` to get a node status in addition to just the name status. This is the flag that provides us with the other uses. Just as we did earlier, we can see that we have a workstation (<00>) and also a file server (<20>). You'll also see from this output, just as we did earlier, that the system belongs to the workgroup WORKGROUP. Workgroups are used for ad hoc Windows networks where there is no domain controller to manage all of the systems.

Using the net Utility

One program that's built into Windows is the `net` utility. This is widely used for several purposes. If you wanted to connect to a shared drive on the network, you would use `net use` to connect to that shared drive. The `net` command allows you to query the network using SMB messages. As an example, the output that follows shows statistics from the workstation service on a Windows server. This shows information about network communication primarily, including bytes transferred. It can also show you the number of sessions that have been started and the sessions that have failed.

```
PS C:\Users\kilroy> net statistics workstation
Workstation Statistics for \\SERVER2020


Statistics since 1/1/2021 6:08:25 PM


  Bytes received                            2994818
  Server Message Blocks (SMBs) received     8
  Bytes transmitted                         5615081
  Server Message Blocks (SMBs) transmitted  0
  Read operations                           1180
  Write operations                          0
  Raw reads denied                          0
  Raw writes denied                         0

  Network errors                            0
  Connections made                          0
  Reconnections made                        0
  Server disconnects                        0

  Sessions started                          0
  Hung sessions                             0
  Failed sessions                           0
  Failed operations                         0
  Use count                                 687
  Failed use count                          0

The command completed successfully.
```

Additionally, you can extract information about the configuration for the system, which will include the computer name, the software version, and the domain the computer has been joined to. One downside to this utility, though, is you need to be on the local network and probably have to be joined to the domain. This is possible if you have already compromised a system and are looking to pivot to another system on the network.

nmap Scripts

`nmap` continues to be relevant to us, even though we've moved beyond the port scanning phase. Here, we are looking to gather more information using the scripts that are provided. At the time of this writing, there are 35 SMB-related scripts included in the implementation of `nmap` on the latest version of Kali Linux. Next, you can see a portion of the output from that script.

The name of the script, `smb-os-discovery`, is shown in the output. This is a Windows system that has been set up for sharing. You'll see that `nmap` has identified it very specifically, down to the service pack that has been installed. Interestingly, there are several other systems on the network where SMB-based sharing is enabled, but none of them get identified. The big difference between those and this one is that those systems have only port 445 open, while this one also has ports 135 and 139 open.

smb-os-discovery Scan Output

```
 $ sudo nmap --script smb-os-discovery 192.168.1.10
 Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-12 13:51
EST

 Nmap scan report for stevedallas.lan (192.168.86.50)
 Host is up (0.00058s latency).

 PORT    STATE SERVICE
 135/tcp open  msrpc
 139/tcp open  netbios-ssn
 445/tcp open  microsoft-ds
 MAC Address: 46:5E:C8:0A:B7:D1 (Unknown)

 Host script results:
 | smb-os-discovery:
 |   OS: Windows 7 Professional 7601 Service Pack 1 (Windows 7
Professional 6.1)
 |   OS CPE: cpe:/o:microsoft:windows_7::sp1:professional
 |   Computer name: stevedallas
 |   NetBIOS computer name: STEVEDALLAS\x00
 |   Workgroup: WORKGROUP\x00
 |_  System time: 2021-01-04T20:30:27-06:00
```

There are several important pieces of information we can use `nmap` to look for. There are enumeration scripts for users, groups, services, processes, and shares. Some of these require authentication before the remote system will give anything up. Microsoft began disabling null session authentication in Windows Server 2008 R2 and Windows 7. Any operating system after that will require authentication before accessing the interprocess communication needed to extract the information requested. However, the setting can be disabled, and you never know when you will run across very outdated or

misconfigured systems. You can see the failure of the share enumeration script in `nmap` here. The listing shows that even though authentication was required, it still attempted common share names.

Enumerating Shares with Nmap

```
 $ sudo nmap --script smb-enum-shares 192.168.1.10
 Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-12 13:53
EST

 Nmap scan report for stevedallas.lan (192.168.86.50)
 Host is up (0.00040s latency).

 PORT    STATE SERVICE
 135/tcp open  msrpc
 139/tcp open  netbios-ssn
 445/tcp open  microsoft-ds
 MAC Address: 46:5E:C8:0A:B7:D1 (Unknown)

 Host script results:
 | smb-enum-shares:
 |   note: ERROR: Enumerating shares failed, guessing at common
ones
 (NT_STATUS_ACCESS_DENIED)
 |   account_used: <blank>
 |   \\192.168.86.50\ADMIN$:
 |     warning: Couldn't get details for share:
NT_STATUS_ACCESS_DENIED
 |     Anonymous access: <none>
 |   \\192.168.86.50\C$:
 |     warning: Couldn't get details for share:
NT_STATUS_ACCESS_DENIED
 |     Anonymous access: <none>
 |   \\192.168.86.50\IPC$:
 |     warning: Couldn't get details for share:
NT_STATUS_ACCESS_DENIED
 |     Anonymous access: READ
 |   \\192.168.86.50\USERS:
 |     warning: Couldn't get details for share:
NT_STATUS_ACCESS_DENIED
 |_    Anonymous access: <none>
```

One of the common share names tried is `IPC$`. This is a share name allowing access to shared pipes, which is a method for interprocess

communications. Another share name `nmap` checked for is `C$`. This is an administrative share that is created. Again, older versions of Windows will allow easier access to this share, but since Windows XP, there have been more restrictions on the use of this share. It does enable administrators to function remotely, but accessing it requires more than just login credentials. The login credentials have to be for an administrator.

While `nmap` does have other scripts that can be used against systems running SMB, most of them are not for enumeration. Many of them are specific to vulnerability identification or confirmation. Since we are focusing on enumeration, we can put off looking at those other scripts until a later time.

NetBIOS Enumerator

An open source tool that will function like `nmap` in the sense that it will scan a system to identify systems that speak SMB before probing them is NetBIOS Enumerator. This is a graphical tool, which makes it easier to locate information more quickly about each system and its capabilities. Figure 6.1 shows the output from a scan of a network. In the output you will see not only systems, including their name and IP address, but also any workgroup or domain that exists on the network.

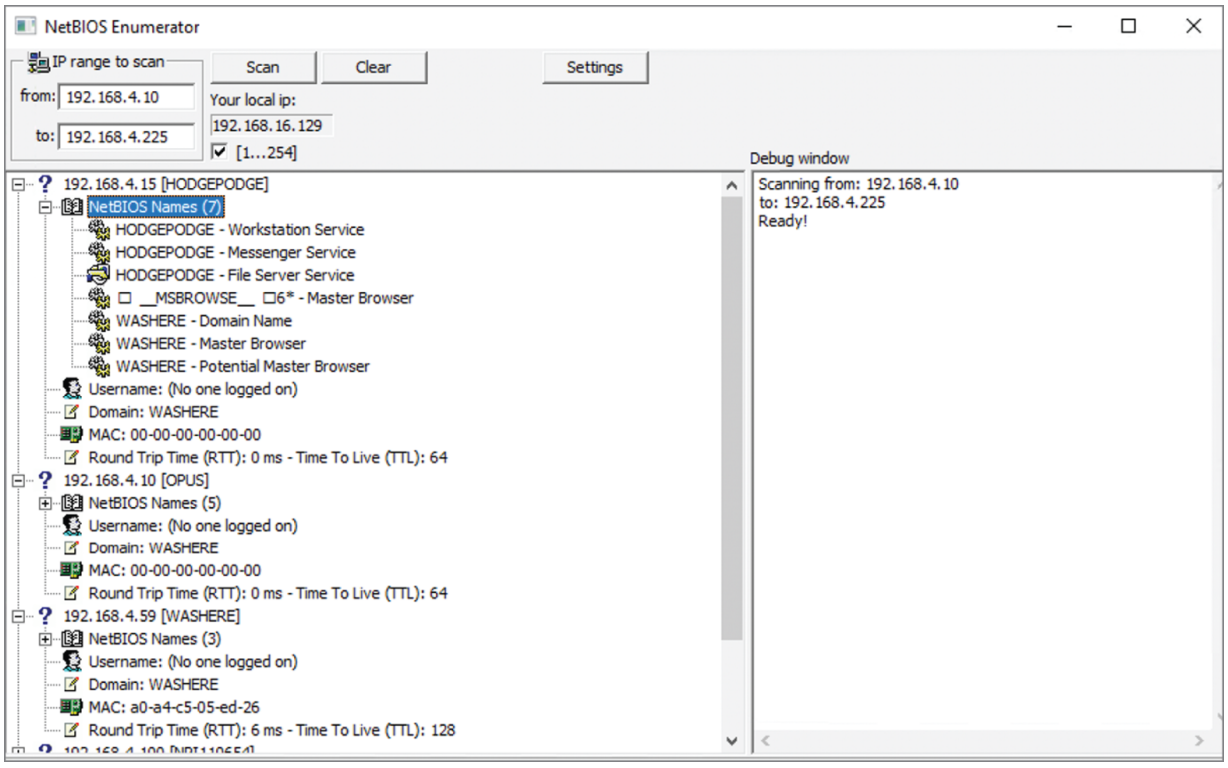

FIGURE 6.1 NetBIOS Enumerator

When running NetBIOS Enumerator, you provide a range of IP addresses, and it will start scanning the network. Once it identifies a system that supports SMB, it starts to query the system to get as much information as it can get. One item that is missing from the output for each of the hosts found is the username of the person logged in. The reason the username is missing is this is an unauthenticated scan. The program has not done any authentication against the remote system, so there is some information that won't be available. SMB is a bit of a promiscuous protocol in the sense that it will provide a lot of information, but it provides only enough information for remote services to work. It doesn't provide everything you can potentially ask for unless you have authenticated.

Metasploit

Metasploit has modules for just about every aspect of ethical hacking. It can be used in so many different ways. As shown earlier, we can definitely use it for enumeration, and when it comes to SMB, there are several that you can run. As an example, we can look for SMB versions across the network. In the following listing, you will see a run of the `smb_version` module. You

should get an idea of the version of the SMB service that's running. What you can see are two systems that have been identified with the operating system version listed. From that information, you can identify the version of SMB that's supported.

The Windows XP system is running SMB version 1 because version 2 didn't come out until Windows Vista was released. According to the SMB version history, the Windows 7 system would be using SMB version 2.1. While it looks like this may be an outdated scan, having old systems around is useful for practicing attacks on, since they are more likely to have vulnerable versions of software.

SMB Version Scan with Metasploit

```
 msf auxiliary(scanner/smb/smb_version)> run

 [*] Scanned 26 of 256 hosts (10% complete)
 [*] 192.168.86.26:445 - Host could not be identified: ()
 [*] 192.168.86.27:445 - Host could not be identified: ()
 [*] 192.168.86.32:445 - Host could not be identified: ()
 [*] 192.168.86.41:445 - Host could not be identified: ()
 [+] 192.168.86.49:445 - Host is running Windows XP SP2
(language:English) (name:OPUS-C765F2) (workgroup:WORKGROUP )
 [+] 192.168.86.50:445 - Host is running Windows 7 Professional
SP1 (build:7601) (name:STEVEDALLAS) (workgroup:WORKGROUP )
 [*] Scanned 52 of 256 hosts (20% complete)
 [*] 192.168.86.61:445 - Host could not be identified: ()
```

We can also use Metasploit to enumerate users. To do that, you would use the `smb_enumusers_domain` module. If you know one, you can use a username and password. This would allow the module to authenticate against the system to obtain additional users. This is not required, though you're much less likely to get a list of users without authentication of some sort. Fortunately, there is another module you can use to help you get at least one username and password. The `smb_login` module can be used to attempt username/password combinations. Here you can see the list of options for the `smb_login` module.

smb_login Module Options

```
 Module options (auxiliary/scanner/smb/smb_login):

            Name                Current Setting  Required
Description
            ----                ---------------  --------  -----
------
    ABORT_ON_LOCKOUT   false            yes       Abort the run
when an account lockout is detected
    BLANK_PASSWORDS    false            no        Try blank
passwords for all users
    BRUTEFORCE_SPEED   5                yes       How fast to
bruteforce, from 0 to 5
    DB_ALL_CREDS       false            no        Try each
user/password couple stored in the current database
    DB_ALL_PASS        false            no        Add all
passwords in the current database to the list
    DB_ALL_USERS       false            no        Add all users
in the current database to the list
    DETECT_ANY_AUTH    false            no        Enable
detection of systems accepting any authentication
    DETECT_ANY_DOMAIN  false            no        Detect if
domain is required for the specified user
    PASS_FILE                           no        File
containing passwords, one per line
    PRESERVE_DOMAINS   true             no        Respect a
username that contains a domain name.
    Proxies                             no        A proxy chain
of format type:host:port[,type:host:port][…]
    RECORD_GUEST       false            no        Record guest-
privileged random logins to the database
    RHOSTS                              yes       The target
address range or CIDR identifier
    RPORT              445              yes       The SMB
service 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
    STOP_ON_SUCCESS    false            yes       Stop guessing
when a credential works for a host
    THREADS            1                yes       The number of
concurrent threads
    USERPASS_FILE                       no        File
containing users and passwords separated by space, one pair per
line
```

```
    USER_AS_PASS       false            no        Try the
username as the password for all users
    USER_FILE                           no        File
containing usernames, one per line
    VERBOSE            true             yes       Whether to
print output for all attempts
```

Using the `smb_login` module, you can provide files that contain usernames and passwords. The module will try to authenticate using the username and password combinations. One thing you will also see is that you can tell the module to try the username as a password. This is commonly disallowed by password policies, but you may run across systems where password policies aren't in place, making it possible for the username and password to be the same.

Another type of enumeration that is useful when it comes to SMB is to look at shares. While shares are common on servers in enterprise environments, users may also enable shares on their desktops, unless it is prohibited administratively. They may do this to easily get files from one user to another. These local shares may have weaker permissions set on them, which may make them easier to get files or other information from. Metasploit has a module for this, as you can see here:

```
msf6> use auxiliary/scanner/smb/smb_enumshares
msf6 auxiliary(scanner/smb/smb_enumshares)> show options

Module options (auxiliary/scanner/smb/smb_enumshares):

   Name               Current Setting    Required  Description
   ----               ---------------    --------  -----------
   HIGHLIGHT_NAME_PA  username|password  yes       PCRE regex
of resource name
   TTERN              |user|pass|Groups           s to
highlight
                      .xml
   LogSpider          3                  no        0 =
disabled, 1 = CSV, 2 =
                                                   table (txt),
3 = one liner
                                                   (txt)
(Accepted: 0, 1, 2, 3
                                                   )
   MaxDepth           999                yes       Max number
```

```
of subdirectorie
                                                  s to spider
   RHOSTS                           yes       The target
host(s), see htt

ps://github.com/rapid7/meta
                                                  sploit-
framework/wiki/Using
                                                  -Metasploit
   SMBDomain          .             no        The Windows
domain to use f
                                                  or
authentication
   SMBPass                          no        The password
for the specif
                                                  ied username
   SMBUser                          no        The username
to authenticat
                                                  e as
   Share                            no        Show only
the specified sha
                                                  re
   ShowFiles          false         yes       Show
detailed information w
                                                  hen
spidering
   SpiderProfiles     true          no        Spider only
user profiles w
                                                  hen share is
a disk share
   SpiderShares       false         no        Spider
shares recursively
   THREADS            1             yes       The number
of concurrent th
                                                  reads (max
one per host)
```

As with `nmap`, Metasploit has multiple modules that can be used against SMB systems. Many of them are related to identifying vulnerabilities. It does provide another tool that you can use to gather information, and the advantage to using Metasploit is that it's backed up with a database where information is stored. You can retrieve host data, services, and other details about the target from the database. This makes it a good record-keeping tool as well as a tool that can be used for enumeration and other forms of scanning.

Other Utilities

Considering the number of devices that use SMB for networking, it's not surprising that there are many tools available for enumerating SMB systems. The program `nbtscan` is one of those. It provides details about systems it finds on the local network, including the NetBIOS name, user, MAC address, and IP address. In the following listing, you can see the output of scanning my home network, identifying every system that has Windows shares available. The scan range has been provided on the command line here, but you can also provide the IP addresses in a file.

Scanning a Network with nbtscan

```
 root@quiche:~# nbtscan 192.168.86.0/24
 Doing NBT name scan for addresses from 192.168.86.0/24

 IP address       NetBIOS Name     Server    User
MAC address
 ------------------------------------------------------------------
--------
 192.168.86.0Sendto failed: Permission denied
 192.168.86.44    NPI110654                 <unknown>
00:00:00:00:00:00
 192.168.86.52    BOBBIE           <server>  BOBBIE
00:00:00:00:00:00
 192.168.86.49    OPUS-C765F2  <server>  <unknown>
00:50:56:3b:ac:3e
 192.168.86.170   MILOBLOOM        <server>  <unknown>
ac:87:a3:1e:6b:30
 192.168.86.50    STEVEDALLAS      <server>  <unknown>
46:5e:c8:0a:b7:d1
 192.168.86.26    YAZPISTACHIO     <server>  <unknown>
f0:18:98:0c:34:69
 192.168.86.61    MILOBLOOM        <server>  <unknown>
ac:87:a3:1e:6b:30
 192.168.86.32    BILLTHECAT       <server>  <unknown>
ac:87:a3:36:d6:aa
 192.168.86.27    BINKLEY          <server>  <unknown>
8c:85:90:5a:7e:f2
 192.168.86.255   Sendto failed: Permission denied
```

Getting the output here, just as with any tool, is helpful, but at some point you need to do something with the information, not least putting it into a

report for your client or employer. One nice thing about `nbtscan` is the ability to generate output that can be manipulated programmatically. This may include taking the output and putting it into a database. While you can certainly read in values with white-space separators, `nbtscan` lets you specify a separator that may make it easier to read in the output, but you can also specify a comma as a separator and then open the output in a spreadsheet program. Adding `-s` followed by whatever character you want to use as a separator will get you the same output as shown earlier, just with your specified separator included between the different fields.

You may start to see a bit of a pattern when it comes to enumeration and SMB. While there are a lot of tools available, they all perform the same functions, and the easiest thing to do when it comes to enumeration is to identify systems that use SMB, including the name they advertise on the network. One note about that name: it may not resolve to an IP address. A name announced using NetBIOS is intended to be used and resolved on the local network. This means it won't resolve using DNS unless DNS is configured to use the same names and IP addresses. It's possible to have one name for your Windows sharing and another one for your DNS, assuming the system even has a DNS address. If your enterprise network uses WINS, they will resolve to be the same because of how the local systems register to WINS.

Another tool, and we'll see the same capabilities with this one, is `enum4linux`. The following example is being run from a Kali Linux system where it is installed by default, but it's easy enough to get a copy of it. It's just a Perl script, so to run it, you need a Perl interpreter. The following example enumerates the shares on a specific host, identified by IP address. The target system is a Linux system running Samba to provide Windows networking functionality over SMB. In the output, you will find a lot of information related to how Samba is configured. As an example, we can see that the workgroup WORKGROUP is configured, which is a way of organizing systems on a local network that are all using Windows.

enum4linux Share Enumeration

```
 root@quiche:~# enum4linux  -S 192.168.86.52
 Starting enum4linux v0.8.9 (
http://labs.portcullis.co.uk/application/enum4linux/ ) on Sun
```

Jan 3 12:18:25 2021

```
 ========================
 |    Target Information    |
 ========================
 Target ……... 192.168.86.52
 RID Range ……... 500-550,1000-1050
 Username …… ''
 Password …… ''
 Known Usernames .. administrator, guest, krbtgt, domain
admins, root, bin, none


 ======================================================
 |    Enumerating Workgroup/Domain on 192.168.86.52    |
 ======================================================
 [+] Got domain/workgroup name: WORKGROUP

 =====================================
 |    Session Check on 192.168.86.52    |
 =====================================
 [+] Server 192.168.86.52 allows sessions using username '',
password ''

 ============================================
 |    Getting domain SID for 192.168.86.52    |
 ============================================
 Domain Name: WASHERE
 Domain Sid: (NULL SID)
 [+] Can't determine if host is part of domain or part of a
workgroup

 ========================================
 |    Share Enumeration on 192.168.86.52    |
 ========================================
 WARNING: The "syslog" option is deprecated

     Sharename       Type      Comment
     ---------       ----      -------
     homes           Disk      Home Directories
     print$          Disk      Printer Drivers
     IPC$            IPC       Service (bobbie server (Samba,
Ubuntu))
 Reconnecting with SMB1 for workgroup listing.

     Server               Comment
     ---------            -------
```

```
    Workgroup            Master
    ---------            -------
    WORKGROUP            STEVEDALLAS

 [+] Attempting to map shares on 192.168.86.52
```

At the bottom, you can see the different share names. This includes the `IPC$`, which is used for interprocess communication between hosts. This allows for management of systems remotely. Using `enum4linux`, we can finally see the master browser on the network. The system named `STEVEDALLAS` holds the current authoritative list of all the systems on the network. This is a system that has been elected by other systems on the network, which essentially means it has volunteered, and based on its characteristics, no one else has challenged it. The reason for `STEVEDALLAS` to be the master browser is it is one of only a couple of Windows systems, and of the two or three that were actual Windows systems and not Linux or macOS running SMB services, `STEVEDALLAS` has the most recent operating system.

Using SMB-related utilities, we can gather a lot of information from the target network. As noted earlier, it's generally necessary to be on the local network to be able to use any of these tools. One reason for this can be the broadcast domain orientation of Windows networking—announcements on the local network. Another is that Windows networking ports are commonly blocked by firewalls. Also, it's entirely possible that desktop networks, where the systems are most likely to be communicating with SMB, often use private addresses that may not be exposed to the outside world. To get to them, because they are nonroutable by convention, you'd have to be on a network where there would at least be routing rules to get to those target networks, meaning there is network reachability.

While desktops are not the only devices that will communicate using SMB, since servers do as well, they are the systems that are going to be the most numerous in most instances. As always, when you are starting to look at the desktops, make sure you have an agreement to look at them with your employer. Not everyone will want the desktop networks to be touched because it can impede productivity for their users. They may also feel like they are most interested in traditional, technical vulnerabilities that are exposed to the outside world, without thinking about lateral movement

within the organization or the fact that the desktops are the most common target with attackers today.

Countermeasures

Because SMB is such an essential foundation of Windows networks, you may think it would be difficult to enact countermeasures to protect against enumeration. However, there are still some things you can do to protect against attackers. Some of this is related to configuration and hardening, while some relates to additional controls you can put into place.

Disable SMBv1 SMBv1 is a much older version of the protocol and is not necessary except in very unusual situations. It is vulnerable to attack in different ways and can be prone to information leakage. This is something you can easily disable. In fact, newer versions of Windows have it disabled by default.

Enable Host-Based Firewall Using zones will help protect individual systems if they are on untrusted networks. A host-based firewall can also have rules that restrict who can talk to individual systems using SMB. In most cases, you shouldn't need to allow SMB conversations originating to desktops except for certain infrastructure systems.

Host-based firewalls can allow you to set up rules restricting who can talk SMB to workstations. The same may be true for Windows servers. You should know which infrastructure systems should allow inbound requests and who they should be allowed from.

Network Firewalls Host-based firewalls will protect individual systems, but often you can catch a lot of this before the host by enabling network firewalls between network segments, only allowing SMB from expected infrastructure systems.

Disable Sharing In enterprise networks, it is generally going to be a bad idea to allow sharing from individual workstations. It's better to put shares on servers where you can control them centrally and let users share files from the file servers. If you disable sharing, you are automatically reducing the information that may be available from outside the host.

Disable NetBIOS over TCP/IP This is not a protocol that should be necessary for most Windows networking functions. This can be disabled, protecting probes of NetBIOS using the TCP/IP stack.

Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) has been around since the late 1980s, though it has gone through several iterations. The first iteration, version 1 (SNMPv1), was introduced in 1988. It continues to be used in spite of being superseded by other versions in the intervening years. SNMPv1 also includes a number of flaws that make it problematic from a security perspective. It is a binary protocol, but there is no encryption supported with it. This means anyone who can obtain the packets being transmitted can decode it easily enough. A second problem is there is very weak authentication. SNMPv1 uses community strings to gain access. You either get read-only access or get read-write access. There is no granularity beyond that. There is also no concept of users. You provide the appropriate string and you get that level of access. Perhaps worse, though, is the fact that the community strings commonly used are so well-known. The string “public” is used for read-only, while the string “private” is used for read-write. This is not hard-coded, but it is common.

Version 2 introduced some fixes to what was widely known as a problematic protocol. First, it introduced enhanced authentication over the basic community string model from v1. However, alongside v2 came v2c, which retained the implementation of the community strings for authentication. This meant that existing tools could continue to just use community strings without having to implement any additional authentication mechanisms. However, v2 and v1 are incompatible. The message specifications in v2 are different from those in v1, so even if your tool was just using community strings, it couldn't just be dropped in place and expected to work with a number of systems using v1.

Version 3 implemented additional fixes and is considered to be a significant improvement over the previous versions. First, it supports encryption rather than plaintext transmissions. Second, it supports user-based authentication. This means you get better accountability to know who is doing what. This

is important, since SNMP can be used not only to monitor devices but also to set parameters on the endpoint.

A basic SNMP architecture would have agents installed on endpoints, while a server system could be used to poll those agents periodically to get measurements for monitoring purposes. An SNMP agent can serve up information that is stored in management information bases (MIBs). These MIBs are defined data structures, using Abstract Syntax Notation One (ASN.1). Each node or data element gets an object identifier (OID), which is a long dotted string of numeric values. Getting or setting any value from the agent requires supplying the correct OID that corresponds with the value you are looking for.

SNMP can supply a lot of different information that is useful if the right MIBs are installed and operating correctly on the agent. This can include things like the system name and the version of the kernel being run on the system. In the following example, you can see the use of the program `snmpwalk` to "walk" the MIB tree to gather data from the agent. This starts at the top level of the tree and gathers what it can find there. From the output of `snmpwalk`, you can see the system name as well as the kernel identifier. Additionally, you can see some contact information that was configured in the SNMP agent. You will see the version number specified with `-v 2c` and the community string with the `-c` parameter. The community string in this case is `'public'`.

snmpwalk of Linux System

```
root@quiche:~# snmpwalk -v 2c -c public 192.168.86.52
iso.3.6.1.2.1.1.1.0 = STRING: "Linux bobbie 4.15.0-30-generic
#32-Ubuntu SMP Sun Jan 4 17:42:43 UTC 2021 x86_64"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (7606508) 21:07:45.08
iso.3.6.1.2.1.1.4.0 = STRING: "Foo <foo@wubble.com>"
iso.3.6.1.2.1.1.5.0 = STRING: "bobbie"
iso.3.6.1.2.1.1.6.0 = STRING: "Erie, CO"
iso.3.6.1.2.1.1.7.0 = INTEGER: 72
iso.3.6.1.2.1.1.8.0 = Timeticks: (1) 0:00:00.01
iso.3.6.1.2.1.1.9.1.2.1 = OID: iso.3.6.1.6.3.11.3.1.1
iso.3.6.1.2.1.1.9.1.2.2 = OID: iso.3.6.1.6.3.15.2.1.1
iso.3.6.1.2.1.1.9.1.2.3 = OID: iso.3.6.1.6.3.10.3.1.1
iso.3.6.1.2.1.1.9.1.2.4 = OID: iso.3.6.1.6.3.1
iso.3.6.1.2.1.1.9.1.2.5 = OID: iso.3.6.1.6.3.16.2.2.1
```

```
iso.3.6.1.2.1.1.9.1.2.6 = OID: iso.3.6.1.2.1.49
iso.3.6.1.2.1.1.9.1.2.7 = OID: iso.3.6.1.2.1.4
iso.3.6.1.2.1.1.9.1.2.8 = OID: iso.3.6.1.2.1.50
iso.3.6.1.2.1.1.9.1.2.9 = OID: iso.3.6.1.6.3.13.3.1.3
iso.3.6.1.2.1.1.9.1.2.10 = OID: iso.3.6.1.2.1.92
```

There are a number of MIBs that can be pulled that will yield essential information. One of these is the interface table, `ifTable`. If you walk the `ifTable`, you will get a list of all of the interfaces on the system and how they are configured. In a tiered network design, you can get an understanding of additional networks the system may be connected to. With this information, you can continue to build out the network map you have been creating as you identify networks and systems.

One advantage to SNMP is that it is perhaps most commonly used on network equipment like routers and switches. As these may not have traditional means of gaining access and may not have more traditional ideas of users, using SNMP to gather information from these devices can be a good entry point. As usual, though, a protocol like SNMP is generally disabled through firewalls. In other words, you would usually have to be allowed specific access to the network device through a firewall or access control list before you could start to request MIB data from a network device. SNMP agents that are properly implemented and configured shouldn't just give out sensitive system information to anyone who asks for it.

Countermeasures

The simplest countermeasure for SNMP is to just disable it. This may not be possible, though, depending on how your network is being managed. Beyond that, always upgrade to SNMPv3 since SNMPv1 has been deprecated for many years and it is not only vulnerable to attack, but also has no possibility to encrypt data, so everything is in plain text. Additionally, SNMPv1 has weak authentication using community strings but implementations very often use well-known words for those community strings. The following should be done to protect your SNMP implementations:

Upgrade to SNMPv3 Get rid of SNMPv1 and SNMPv2.

Require Authentication Disable the use of just community strings for authentication.

Require Encryption Be sure all SNMP communications are encrypted.

Use Firewalls Use a combination of network- and host-based firewalls to guarantee only the right hosts communicate with one another. SNMP should not be allowed without restriction across the network.

Simple Mail Transfer Protocol

Like so many other protocols, SMTP operates using a series of verbs to interact with the server. The client sends a verb and any other necessary parameters to the SMTP server. Based on the verb, the server knows how to handle the parameters received. Unlike other, simpler protocols, though, communicating with SMTP is an entire conversation. Before you start, you have to greet the server. This tells the server what flavor of SMTP you are going to be speaking. You then tell the server what you want to do. Based on the function you are trying to perform, you may have to provide additional information. This may include providing credentials. You can see an example of a simple SMTP conversation next. This is entirely manual, so you can see the conversation at the protocol level and how you might interact with an SMTP server. You may get errors if you aren't connecting to a local server or to a user the SMTP server knows about.

SMTP Conversation

```
root@quiche:~# nc 192.168.86.52 25
220 bobbie.lan ESMTP Postfix (Ubuntu)
EHLO blah.com
250-bobbie.lan
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
```

```
250 SMTPUTF8
MAIL From: foo@foo.com
250 2.1.0 Ok
RCPT To: wubble@wubble.com
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
From: Goober
To: Someone
Date: today
Subject: Hi

Nothing really to say.

.
250 2.0.0 Ok: queued as 33471301389
```

Once you initiate the conversation using either `HELO` or `EHLO`, you will get a list of capabilities offered by the server. There are a couple of capabilities in SMTP that we can use to enumerate users or at least email addresses. One of them you can see is `VRFY`, which can be used to verify users. Not all mail servers will have this feature enabled, since it can be used to identify legitimate users or email addresses. That means it can be used by attackers as well as spammers. Here you can see an example of the use of `VRFY` against a local mail server running Postfix, which has `VRFY` enabled by default.

Testing VRFY

```
root@quiche:~# nc 192.168.86.52 25
220 bobbie.lan ESMTP Postfix (Ubuntu)
EHLO blah.com
250-bobbie.lan
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250 SMTPUTF8
VRFY root@localhost
```

```
 252 2.0.0 root@localhost
 VRFY kilroy@localhost
 252 2.0.0 kilroy@localhost
 VRFY root
 252 2.0.0 root
```

We don't get anything back from our attempts except a status code. There is no text indicating what the server thinks of our request. That means we need to look up the numeric value we get. Unlike the earlier case, where we got 250, which means success, this time we got a status 252. This means that the address can't be verified, but the server will attempt to deliver the message. While `VRFY` is enabled on this server, we don't get a lot of useful information. On top of that, running through this manually is very time-consuming.

We could do it manually. Metasploit again to the rescue, though. The module `smtp_enum` will take a word list and do the same thing automatically that you saw done manually earlier. It will run through all the users in the word list, checking to see whether each user exists. There are two ways to test whether users exist—either the `VRFY` command or the `MAIL TO` command. In the following listing, you can see the results of a run against the same server. This is using the default word list that comes with Metasploit that has a list of common Unix usernames (`unix_users.txt`).

smtp_enum Run

```
 msf auxiliary(scanner/smtp/smtp_enum)> use
auxiliary/scanner/smtp/smtp_enum
 msf auxiliary(scanner/smtp/smtp_enum)> show options

 Module options (auxiliary/scanner/smtp/smtp_enum):

    Name       Current Setting       Required  Description
    ----       ---------------       --------  -----------
    RHOSTS                           yes       The target
host(s), see https://

github.com/rapid7/metasploit-fra

mework/wiki/Using-Metasploit
    RPORT      25                    yes       The target port
(TCP)
```

```
    THREADS    1                       yes       The number of
concurrent threads
                                                 (max one per
host)
    UNIXONLY   true                    yes       Skip Microsoft
bannered servers
                                                 when testing
unix users
    USER_FILE  /usr/share/metasploi    yes       The file that
contains a list of
               t-framework/data/wor              probable users
accounts.
               dlists/unix_users.tx
               t


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

 msf auxiliary(scanner/smtp/smtp_enum)> set RHOSTS
192.168.86.52/32
 RHOSTS => 192.168.86.52/32
 msf auxiliary(scanner/smtp/smtp_enum)> run

 [*] 192.168.86.52:25 - 192.168.86.52:25 Banner: 220 bobbie.lan
ESMTP Postfix (Ubuntu)
 [+] 192.168.86.52:25 - 192.168.86.52:25 Users found:, backup,
bin, daemon, games, gnats, irc, list, lp, mail, man,
messagebus, news, nobody, postmaster, proxy, sshd, sync, sys,
syslog, uucp, www-data
 [*] Scanned 1 of 1 hosts (100% complete)
 [*] Auxiliary module execution completed
```

You'll notice that there are a lot of users listed as being found by this module. Based on the fact that the `VRFY` command returned a 252 and the users are ones that exist on this system, the module isn't using `VRFY`. Instead, it's using the `MAIL TO` command. Users that don't exist will result in a 550 status code. Users that do exist will return a 250 when mail is attempted to that user. Based on the results of this, the module returns valid usernames, since out of the 112 users in the list, only 21 users were identified.

There is another command that can be used on SMTP servers, though it is targeted at mailing lists. If you find a mailing list address that belongs to a domain, you may be able to use `EXPN` to expand the mailing list, which

means identifying the email addresses that are on that mailing list. This function, though, requires that the server support enhanced SMTP (ESMTP). You can test whether a server supports ESMTP by checking to see whether it accepts `EHLO`, which is the ESMTP version of `HELO`.

Countermeasures

When it comes to countermeasures for SMTP enumeration, we are mostly looking to try to protect against identifying users by probing the mail server. Given the prevalence of username lists, this protection may be of limited use, but it's worth adding in any protection that may make things a little harder.

Disable VRFY Some SMTP servers will allow you to disable this command. Removing it from your mail server should not impact any functionality required for the delivery of mail.

Ignore Unknown Addresses Have your SMTP server ignore usernames/email addresses that are not known. Generating an error provides the attacker with information about the existence or nonexistence of the user.

Restrict Information in Headers Be sure you are not showing information about your internal email chain to the extent you can. This will provide information to the attacker about internal systems, as well as other SMTP servers within your environment.

Disable Open Relays This is far less common than it used to be but ensuring all systems in your environment are not allowing relaying to other domains will help protect your systems from being used to enumerate other domains, since a lot of end-user networks, such as those belonging to cable networks, restrict outbound SMTP to everything other than the cable provider's email address.

Implement Email Security While these practices may not be perfect at protecting against enumeration, the more enterprises that implement practices like the Sender Policy Framework (SPF) and DomainKeys Identified Mail (DKIM), the better we will be able to restrict who can and can't send email, which includes probing servers for email addresses.

Web-Based Enumeration

As far as enumeration goes, there may be a couple of things we want to look at on web servers. The first is to identify directories available in a website. There are a lot of different ways to do this, especially if you have access to web application testing tools. Even if you don't or aren't familiar with using them, there are simple ways of checking. All you need is a word list that can provide potential directory names and a tool that can make requests to a web server based on those words—appending each word to a base uniform resource locator (URL).

Here, you can see the use of the program `dirb`, which includes its own word list of common directory names. This was run against a web server on my own network that had the WordPress distribution unzipped into the base web directory.

dirb Directory Testing

```
root@quiche:~# dirb http://192.168.86.52/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sun Dec 3 19:38:36 2022
URL_BASE: http://192.168.86.52/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.86.52/ ----
+ http://192.168.86.52/index.php (CODE:200|SIZE:418)
==> DIRECTORY: http://192.168.86.52/wp-admin/
==> DIRECTORY: http://192.168.86.52/wp-content/
==> DIRECTORY: http://192.168.86.52/wp-includes/
+ http://192.168.86.52/xmlrpc.php (CODE:200|SIZE:3065)

---- Entering directory: http://192.168.86.52/wp-admin/ ----
+ http://192.168.86.52/wp-admin/admin.php
(CODE:200|SIZE:10531)
```

```
 ==> DIRECTORY: http://192.168.86.52/wp-admin/css/
 ==> DIRECTORY: http://192.168.86.52/wp-admin/images/
 ==> DIRECTORY: http://192.168.86.52/wp-admin/includes/
 + http://192.168.86.52/wp-admin/index.php (CODE:200|SIZE:7265)
 ==> DIRECTORY: http://192.168.86.52/wp-admin/js/
 ==> DIRECTORY: http://192.168.86.52/wp-admin/maint/
 ==> DIRECTORY: http://192.168.86.52/wp-admin/network/
 ==> DIRECTORY: http://192.168.86.52/wp-admin/user/
```

What we've done so far is to check known or expected directories on a web server. Using a word list doesn't guarantee that you are going to identify all directories that are available on the server. If a directory isn't in the word list, it won't be identified. We can turn to another tool to help with fuzzing directory names, meaning generating names dynamically based on a set of rules. You may expect at this point that we would turn to Metasploit because it's so useful. You'd be correct. We can use the `brute_dirs` module. Using this module, you set a format for what a directory name could or should look like and the module will run through all possible names that match the format. Here you can see the options available for the module, followed by a format set. We're going to be testing against all words with lowercase characters whose lengths are between one and eight characters.

brute_dirs Metasploit Module

```
 msf> use auxiliary/scanner/http/brute_dirs
 msf auxiliary(scanner/http/brute_dirs)> info

        Name: HTTP Directory Brute Force Scanner
      Module: auxiliary/scanner/http/brute_dirs
    License: BSD License
        Rank: Normal

 Provided by:
   et <et@metasploit.com>

 Basic options:
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   FORMAT   a,aa,aaa         yes       The expected directory
format (a alpha, d digit, A upperalpha)
   PATH     /                yes       The path to identify
directories
   Proxies                   no        A proxy chain of format
```

```
type:host:port[,type:host:port][…]
   RHOSTS   192.168.86.52    yes       The target address range
or CIDR identifier
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for
outgoing connections
   THREADS  1                yes       The number of concurrent
threads
   VHOST                     no        HTTP server virtual host

 Description:
   This module identifies the existence of interesting
directories by
   brute forcing the name in a given directory path.

 msf auxiliary(scanner/http/brute_dirs)> set FORMAT
a,aa,aaa,aaaa,aaaaa,aaaaaa,
 aaaaaaa,aaaaaaaa
 FORMAT => a,aa,aaa,aaaa,aaaaa,aaaaaa,aaaaaaa,aaaaaaaa
 msf auxiliary(scanner/http/brute_dirs)> run

 [*] Using code '404' as not found.
```

Metasploit, as always, has a large number of modules that can be used for web-based enumeration beyond just identifying directories on the web server. As you start working with websites and, more specifically, web applications, you will run across a lot of open source applications that are well-known because they are so commonly used. As an example, you may find a WordPress installation. Again using Metasploit, we can enumerate the users in the WordPress installation. The `wordpress_login_enum` module can take a user file or a password file, or you could provide a single username with a password file or a single password with a username file. There are a number of other options that can be set in the module, providing a lot of capabilities. Here you can see running the module against a local installation of WordPress.

Enumerating Usernames in WordPress

```
 msf auxiliary(scanner/http/wordpress_login_enum)> set
BLANK_PASSWORDS true
 BLANK_PASSWORDS => true
 msf auxiliary(scanner/http/wordpress_login_enum)> set RHOSTS
192.168.86.52
```

```
 RHOSTS => 192.168.86.52
 msf auxiliary(scanner/http/wordpress_login_enum)> run

 [*] / - WordPress Version 4.9.8 detected
 [*] 192.168.86.52:80 - / - WordPress User-Enumeration -
Running User Enumeration
 [+] / - Found user 'kilroy' with id 1
 [+] / - Usernames stored in:
 /root/.msf4/loot/20210104205530_default_192.168.86.52_wordpres
s.users_790698.txt
 [*] 192.168.86.52:80 - / - WordPress User-Validation - Running
User Validation
 [*] 192.168.86.52:80 - [1/0] - / - WordPress Brute Force -
Running Bruteforce
 [*] / - Brute-forcing previously found accounts…
 [*] Scanned 1 of 1 hosts (100% complete)
 [*] Auxiliary module execution completed
```

You'll notice it makes reference to storing the username in a file in the home directory of the root user, which is the user under which `msfconsole` is running. Metasploit also stores this information. Anytime you want to check to see what you have grabbed in terms of information like credentials, you can run the command `loot` inside `msfconsole`. You can see the results of this command here.

Listing loot in msfconsole

```
 msf auxiliary(scanner/http/wordpress_login_enum)> loot

 Loot
 ====

 host         service  type         name               content
info     path
 ----         -------  ----         ----               -------
----     ----
 192.168.86.52           wordpress.users
192.168.86.52_wordpress_users.txt  text/plain
 /root/.msf4/loot/20210104205530_default_192.168.86.52_wordpres
s.users_790698.txt
```

When it comes to WordPress, we don't have to rely on Metasploit. Again, we can rely on Kali Linux because it's freely available and easy to use, not

to mention that there are hundreds of tools that are available. Kali comes with the program `wpscan` that can be used to enumerate not only users but also themes and plugins. When it comes to a web application like WordPress, the plugins can also be useful to know about because they may also introduce vulnerabilities. They do include additional code after all. In the following listing, you can see a run of `wpscan`, where we enumerate the plugins. You will also notice that while it was running, it detected the user that is configured.

Enumerating Plugins in WordPress

```
$ wpscan --url http://192.168.1.15
_______________________________________________________________
_
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 3.8.22
      Sponsored by Automattic - https://automattic.com/
      @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________
_

 Scan Aborted: The remote website is up, but does not seem to
be running WordPress.

┌──(kilroy㉿badmilo)-[~]
└─$ wpscan --url http://192.168.1.15
_______________________________________________________________
_
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
```

```
                         Version 3.8.22
       Sponsored by Automattic - https://automattic.com/
       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________
_

 [+] URL: http://192.168.1.15/ [192.168.1.15]
 [+] Started: Sun Dec 4 14:33:09 2022

 Interesting Finding(s):

 [+] Headers
  | Interesting Entry: Server: Apache/2.4.52 (Ubuntu)
  | Found By: Headers (Passive Detection)
  | Confidence: 100%

 [+] XML-RPC seems to be enabled:
http://192.168.1.15/xmlrpc.php
  | Found By: Direct Access (Aggressive Detection)
  | Confidence: 100%
  | References:
  |  - http://codex.wordpress.org/XML-RPC_Pingback_API
  |  -
https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpr
ess_ghost_scanner/
  |  -
https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_
xmlrpc_dos/
  |  -
https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpr
ess_xmlrpc_login/
  |  -
https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpr
ess_pingback_access/

 [+] WordPress readme found: http://192.168.1.15/readme.html
  | Found By: Direct Access (Aggressive Detection)
  | Confidence: 100%

 [+] Upload directory has listing enabled:
http://192.168.1.15/wp-content/uploads/
  | Found By: Direct Access (Aggressive Detection)
  | Confidence: 100%

 [+] The external WP-Cron seems to be enabled:
http://192.168.1.15/wp-cron.php
  | Found By: Direct Access (Aggressive Detection)
  | Confidence: 60%
```

```
  | References:
  |  - https://www.iplocation.net/defend-wordpress-from-ddos
  |  - https://github.com/wpscanteam/wpscan/issues/1299

 [+] WordPress version 6.1.1 identified (Latest, released on
0001-01-01).
  | Found By: Rss Generator (Passive Detection)
  |  - http://192.168.1.15/index.php/feed/,
<generator>https://wordpress.org/?v=6.1.1</generator>
  |  - http://192.168.1.15/index.php/comments/feed/,
<generator>https://wordpress.org/?v=6.1.1</generator>

 [+] WordPress theme in use: twentytwentythree
  | Location: http://192.168.1.15/wp-
content/themes/twentytwentythree/
  | Readme: http://192.168.1.15/wp-
content/themes/twentytwentythree/readme.txt
  | [!] Directory listing is enabled
  | Style URL: http://192.168.1.15/wp-
content/themes/twentytwentythree/style.css
  | Style Name: Twenty Twenty-Three
  | Style URI: https://wordpress.org/themes/twentytwentythree
  | Description: Twenty Twenty-Three is designed to take
advantage of the new design tools introduced in WordPress 6….
  | Author: the WordPress team
  | Author URI: https://wordpress.org
  |
  | Found By: Urls In Homepage (Passive Detection)
  |
  | Version: 1.0 (80% confidence)
  | Found By: Style (Passive Detection)
  |  - http://192.168.1.15/wp-
content/themes/twentytwentythree/style.css, Match: 'Version:
1.0'

 [+] Enumerating All Plugins (via Passive Methods)
 [+] Checking Plugin Versions (via Passive and Aggressive
Methods)

 [i] Plugin(s) Identified:

 [+] gutenberg
  | Location: http://192.168.1.15/wp-content/plugins/gutenberg/
  | Latest Version: 14.6.1 (up to date)
  | Last Updated: 2022-11-25T18:07:00.000Z
  |
  | Found By: Urls In Homepage (Passive Detection)
  |
```

```
  | Version: 14.6.1 (90% confidence)
  | Found By: Change Log (Aggressive Detection)
  |  - http://192.168.1.15/wp-
content/plugins/gutenberg/changelog.txt, Match: '= 14.6.1'

 [+] Enumerating Config Backups (via Passive and Aggressive
Methods)
  Checking Config Backups - Time: 00:00:00 <=> (137 / 137)
100.00% Time: 00:00:00

 [i] No Config Backups Found.

 [!] No WPScan API Token given, as a result vulnerability data
has not been output.
 [!] You can get a free API token with 25 daily requests by
registering at https://wpscan.com/register

 [+] Finished: Sun Dec 4 14:33:14 2022
 [+] Requests Done: 172
 [+] Cached Requests: 5
 [+] Data Sent: 42.973 KB
 [+] Data Received: 2.096 MB
 [+] Memory used: 240.375 MB
 [+] Elapsed time: 00:00:05In addition to user and plugin
enumeration, wpscan identified a couple of issues with the
WordPress installation, so those can be used down the road. It
also identified a header from the HTTP communication that it
felt to be interesting because it included the name of the
product as well as the version and the operating system. All of
these are useful pieces of information to have.
```

These aren't the only things we can enumerate when it comes to web applications. We could scan networks for different web applications or look to enumerate users. Looking for directories that are on the web server is also useful because it can help us identify applications as well as data that may be available.

Countermeasures

There is a lot of information that can be leaked from web servers. There are some techniques you can use to protect against web-based enumeration, though.

Restrict Information Provided Headers and error messages should not provide information about the server and especially version numbers, including information about the underlying operating system or application stack.

Use Appropriate Access Control Ensure all web applications have strong authentication enabled and appropriate access controls to protect resources from unauthorized users.

Disable Directory Listings Disable open directory listings from web servers to protect potentially sensitive files from open access.

Summary

Enumeration is the process of gathering a lot of information further up the network stack than just IP addresses and ports. At this point, we are moving up to the Application layer. We're looking for things like usernames, where we can find them, and network shares and any other footholds we may be able to gather. To accomplish this enumeration work, there are a number of protocols and tools that we can use. The first is `nmap`, because we need to go beyond just identifying open ports. We need to identify the services that are in use, including the software being used. One feature of `nmap` that is useful, especially in these circumstances, is its scripting capability. This includes, especially, all the scripts that are built into `nmap`.

When it comes to `nmap`, there are scripts that can be used not only to probe services for additional details but also to take advantage of the many enumeration capabilities. One of the protocols we can spend time looking at is the SMB protocol. `nmap` includes a number of scripts that will probe systems that use SMB. This includes identifying shares that may be open as well as, potentially, users and other management-related information that can be accessed using SMB.

SMB relies on RPCs. NFS, a file sharing protocol developed by Sun Microsystems, also uses RPC. We can use `nmap` to enumerate RPC services, since these services register dynamically with a mapping or registry service. Probing the RPC server will provide details about the programs and ports that are exporting RPC functionality. If the program is written in Java, it will use RMI instead of `portmap` or the SunRPC protocol.

Another program you can use across a number of protocols for enumeration is Metasploit. Metasploit comes with lots of modules that will enumerate shares and users on SMB, services using SunRPC, and a number of other protocols. If there is information that can be enumerated, Metasploit probably has a module that can be run. This includes modules that will enumerate users in mail servers over SMTP. You can also enumerate information using SNMP. Of course, when it comes to SNMP, you can also use tools like `snmpwalk`.

While Metasploit can be used across a lot of different protocols to look for different pieces of useful information, it is not the only tool you can use. There are built-in tools for gathering information from SMB, for example. You're more likely to find those tools on Windows systems, but you can also find tools on Linux systems, especially if you have Samba installed. Samba is a software package that implements the SMB protocol on Unix-like systems. There are also a lot of open source tools that can be used for different protocols. If you are OK with using Linux, Kali Linux is a distribution that includes hundreds of security-related tools.

As you are performing this enumeration, you should be taking notes so you have references when you are going forward. One advantage to using Metasploit, not to oversell this software, is that it uses a database backend, which will store a lot of information automatically. This is certainly true of services and ports but also of usernames that have been identified. This is not to say that Metasploit can be used to store every aspect of your engagement, but you can refer to details later by querying the Metasploit database as needed.

Every service that can be enumerated should have countermeasures deployed to protect those services. Each service will have different countermeasures, though in general, restricting who has access to the different services, to the extent possible, will help protect the service from having information gathered easily.

Review Questions

You can find the answers in the appendix.

1. What are RPCs primarily used for?

A. Interprocess communications

B. Interprocess semaphores

C. Remote method invocation

D. Process demand paging

2. What would you be trying to enumerate if you were to use enum4linux?

A. Procedures

B. Linux-based services

C. Shares and/or users

D. Memory utilization

3. How do you authenticate with SNMPv1?

A. Username/password

B. Hash

C. Public string

D. Community string

4. What SMTP command would you use to get the list of users in a mailing list?

A. `EXPD`

B. `VRFY`

C. `EXPN`

D. `VRML`

5. What type of enumeration would you use the utility `dirb` for?

A. Directory listings

B. Directory enumeration

C. Brute-force dialing

D. User directory analysis

6. What are data descriptions in SNMP called?

A. Management-based information

B. Data structure definition

C. Extensible markup language

D. Management information base

7. What is the process Java programs identify themselves to if they are sharing procedures over the network?

   A. RMI registry

   B. RMI mapper

   C. RMI database

   D. RMI process

8. You are working with a colleague, and you see them interacting with an email server using the `VRFY` command. What is it your colleague is doing?

   A. Verifying SMTP commands

   B. Verifying mailing lists

   C. Verifying email addresses

   D. Verifying the server config

9. What is the SMB protocol used for?

   A. Data transfers using NFS

   B. Data transfers on Windows systems

   C. Data transfers for email attachments

   D. Data transfers for Windows Registry updates

10. Which of these is a built-in program on Windows for gathering information using SMB?

    A. `nmblookup`

    B. `smbclient`

    C. Metasploit

D. `nbtstat`

11. What status code will you get if your attempt to use the `VRFY` command fails?

    A. 550

    B. 501

    C. 250

    D. 200

12. What program would you use to enumerate services?

    A. `smbclient`

    B. `nmap`

    C. `enum4linux`

    D. `snmpwalk`

13. Which version of SNMP introduced encryption and user-based authentication?

    A. 1

    B. 2

    C. 2c

    D. 3

14. Which of these could you enumerate on a WordPress site using `wpscan`?

    A. Plugins

    B. Posts

    C. Administrators

    D. Versions

15. Which of these tools allows you to create your own enumeration function based on ports being identified as open?

    A. Metasploit

B. `nmap`

C. Netcat

D. `nbtstat`

16. What underlying functionality is necessary to enable Windows file sharing?

    A. Network File System

    B. Common Internet File System

    C. Remote procedure call

    D. Remote method invocation

17. What is the `IPC$` share used for?

    A. Process piping

    B. Interprocess construction

    C. Remote process management

    D. Interprocess communication

18. What tool does a Java program need to use to implement remote process communication?

    A. JRE

    B. `rmic`

    C. `rmir`

    D. JDK

19. Which of these passes objects between systems?

    A. SunRPC

    B. SMB

    C. RMI

    D. `nmap`

20. If you needed to enumerate data across multiple services and also store the data for retrieval later, what tool would you use?

A. Metasploit

B. `nmap`

C. RMI

D. Postgres

21. Which SMTP command would be easiest to disable to prevent attackers from misusing it for enumeration?

    A. `MAIL FROM`

    B. `RCPT TO`

    C. `DATA`

    D. `VRFY`

22. Which version of SNMP should network administrators be running?

    A. V1

    B. V2

    C. V3

    D. V4

23. Which of these is not a way to protect against enumeration with SMB?

    A. Implement the latest NetBIOS patches

    B. Disable SMB

    C. Use network firewalls

    D. Disable sharing

24. Which Metasploit module would you use to take advantage of potentially weaker permissions on an end user's workstation?

    A. auxiliary/scanner/smb/smb_enumusers_domain

    B. auxiliary/admin/smb/delete_file

    C. auxiliary/admin/smb/check_dir_file

    D. auxiliary/scanner/smb/smb_enumshares

25. Which security technology could you use to protect against enumeration on services?

A. Intrusion detection system (IDS)

B. Host-based firewall

C. Endpoint Detection and Response (EDR)

D. Anti-malware