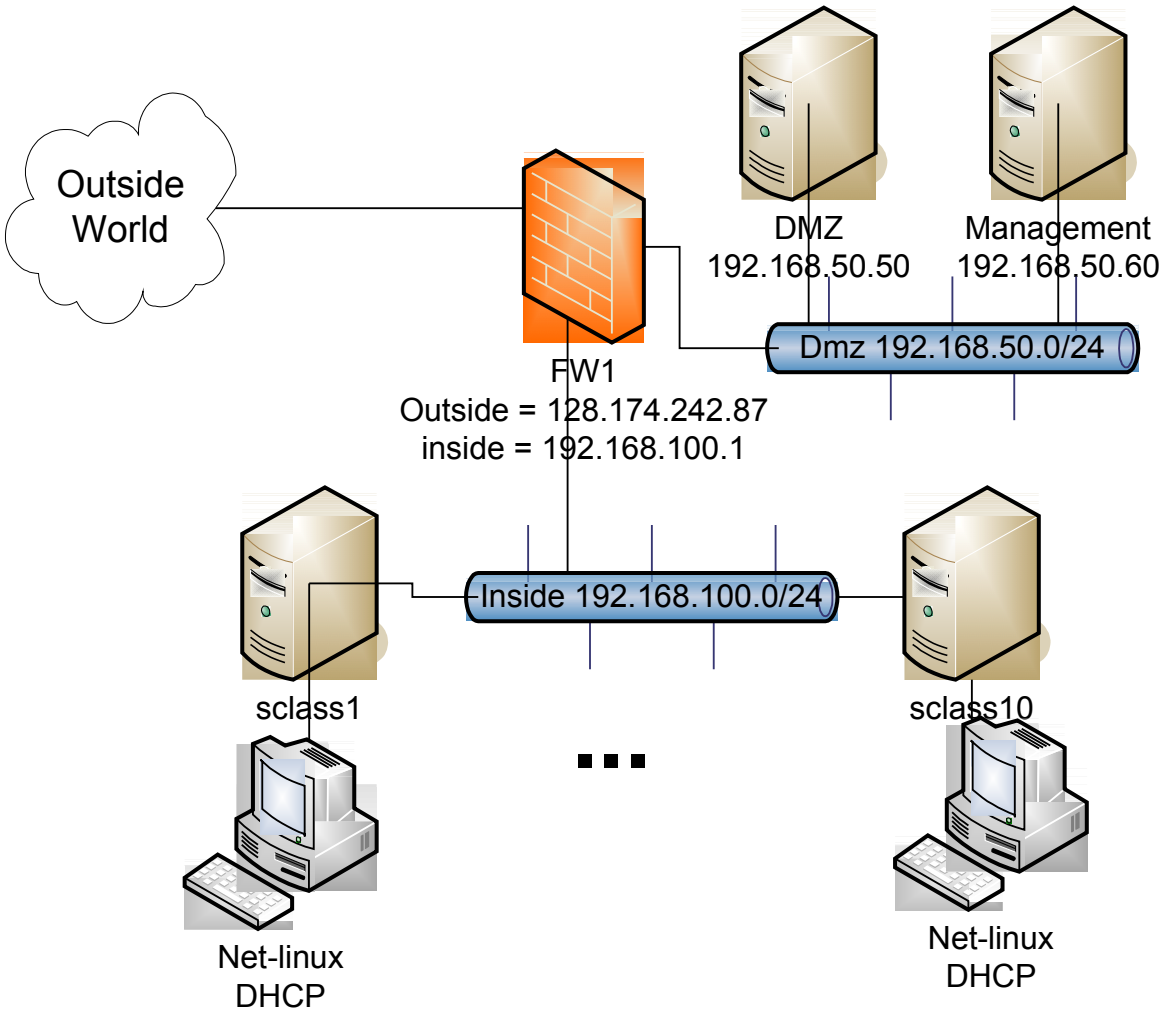
Notes on Running Dsniff and the Lab Network Environment

Cyber Security Lab
2/16/10

1 General Overview

1.1 Initial machine configuration



The lab has 10 dell systems. Each dell computer has Fedora Core 12 (FC12) and VirtualBox installed. The host machine is named sclassX (where X is a value between 1 and 10). The machines use DHCP to retrieve their IP addresses and DNS information from FW 1, an IOS firewall. The firewall enables connections initiated from the lab to

the outside world, but no connections initiated from the outside world in. The connections between the machines, the firewall, and the outside world are made through two VLAN's implemented on the 24 port switch.

Port 1 is currently configured as a span port for traffic on the inside vlan. I'll try to have a machine plugged into that port running wireshark during the lab period.

There are two older machines on a desk next to the network rack (DMZ and management). The Management machine is used to display the console to access the networking equipment (more on that later). DMZ hosts a very simple web server and ftp server.

1.2 Determining Addresses on Linux

From a shell window, you can use the "ifconfig" command to see all the addresses associated with the interfaces. This is in the /sbin directory (in case it is not on your path). You can call "ifdown eth0" and "ifup eth0" to retrigger the DHCP request. Or you can call "dhclient" to restart the address negotiation.

If you use the "-" option for su, you will take on the environment of the target user. Otherwise, you will maintain your original environment variables. So if you execute "su -", you will get the path of the root user, and /sbin will be on your path. If you execute "su", you will maintain the path of the original user (which probably does not include /sbin and /usr/sbin..

The net-linux VM should be set up as a bridged adapter with an address in the 192.168.100.0/24 network. If its address is not in that network, the adapter is in NAT'ed and needs to be changed in the VirtualBox configuration.

2 Tools to Try

2.1 Packet Sniffers

The virtual linux machines and most of the base linux machines have the sniffer wireshark installed. Invoking *wireshark* will bring up a screen. Use the capture menu to start capturing traffic. You can configure capture to show packets as they come in, or you can wait and look at the packets after you stop the capture.

The top grid shows a summary of all the captured packets. The bottom window shows the details of the currently selected packet. Try identifying packets of interest (or lack of interest like the loopback or spanning tree traffic) and set filters to constrain the set of viewable packets. Right click on a field identifying the packet of interest (or lack of interest) and select one of the *add filter* options.

2.2 Dsniff Package

Dsniff is a package of sniffing and simple attack tools written around 1999/2000. Some of the attacks deal with now obsolete versions of protocols, so they may be difficult to get working.

Successfully launching the dsniff attacks requires coordination. For example, machines cannot mutually arpspoof each other. Pair up with people working on another machine and try to set up some of these scenarios together.

2.3 ArpSpoof

You can use *arpspoof* poison a target's ARP cache, so it will use the attacker's MAC address. This will route all traffic from the victim machine through attacking machine enabling sniffing which is not normally available in a switched environment.

The following command

arpspoof -i eth2 –t 192.168.100.16 192.168.100.1

will poison the arp cache of the machine with the address 192.168.100.16, to it will think the attacker machine is the default gateway (192.168.100.1). The arp cache clears pretty quickly (in a couple seconds), so you must keep the arpspoof program running during your attack.

The attacker machine must turn on IP Forwarding. This can be controlled through the proc pseudo file system. Change the contents of the file /proc/sys/net/ipv4/ip_forward from 0 to 1. You also want to make sure that that host firewall is down. Run "/sbin/service iptables stop" to turn off the firewall immediately.

To ensure that the return traffic routes through the attacker you will also need to poison the gateway's cache.

Now traffic from 192.168.100.16 should pass to the outside world and return traffic should come back. Both the forward and the return traffic will pass through the attacking machine. Use the packet sniffer on the attacking machine to verify this traffic flow.

Use the "arp -a" command to check the current state of the arp cache on the target.

2.4 Dnsspoof

Arpspoof must be running to ensure that the DNS requests passes through the attacking machine. Or the machine must be sitting on a span port. Dnsspoof will intercept all DNS requests and reply with the attacking address. Look at the man page. You can specify hosts files to only respond to some dns requests. The default is to respond to all requests with the address of the current machine.

You can see the results of the dnsspoof by doing a wget from the victim machine, e.g. "wget microsoft.com" should show that it is really requesting files from the attacking machine.

2.5 Dsniff

The dsniff program is a password sniffer. It looks for cleartext passwords in the traffic stream. You should use this in conjunction to arpspoof to get all traffic through the attacking machine for sniffing.

I've successfully used dsniff to pull a ftp password in the past. Try using dsniff with the -c option if it doesn't seem to be pulling out the passwords. The DMZ should have a ftp server installed, so you can try logging in as one of the standard users.

2.6 MITM tools

At this point it is obvious how you could write man in the middle (MITM) tools. The Dsniff package includes several MITM tools. These MITM tools assume that you are also running arpspoof and dnsspoof.

I finally had some success with webmitm last year. Webmitm uses openssl to help you create a certificate. It then intercepts web traffic and passes it onto the ultimate destination. For non-SSL traffic it will pass traffic and print out passwords. For SSL traffic it will present its certificate. The browser will print many warning messages. If you click through them it will show you the first SSL page. In past years, my SSL session would stall out after the first attempt. This year, I was able to login and look at a few pages, but when I tried to restart the session, I had some problems, so it still appears to be a bit sensitive. Alternatively my poor VM might have been a bit overloaded.

I have a user temporarily set up a login on https://network-geographics.com that you can use to test logging in over a SSL connection. User=sclass PW=class-test.

There is also a sshmitm that works on the same principle. It has the potential to be much harder to detect, because SSH certificates are generally self signed. In a fresh, default environment, it would be very hard to detect. Sshmitm only support SSH version 1, as a conscious attempt to limit its usefulness to the attacker.

2.7 Other Dsniff Tools

There are a number of other tools in the Dsniff suite that take advantage of the fact that all traffic from a third party is passing through your machine. Try out some of the following:

- filesnarf
- macof
- mailsnarf
- msgsnarf
  sshow
- tcpkill
- tcpnice
- urlsnarf
- webspy