

Notes on running Dsniff in the lab

Cyber Security Lab
1/22/08

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.

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.

ArpSpoof

You can use *arp spoof* 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

```
arp spoof -t 192.168.50.16 192.168.50.1
```

will poison the arp cache of the machine with the address 192.168.50.16, to it will think the attacker machine is the default gateway (192.168.50.1). 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 do not need to explicitly poison the gateway in our situation. It use the MAC address from the initiating packet (which will be the attacker's MAC) when filling in its session table. So the reverse traffic will pull the attacker's MAC from the session entry.

In other situations, you would need to run arpspoof a second time to poison the gateway's arp cache.

Now traffic from 192.168.50.16 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 "arp -a" command to check the current state of the arp cache on the target.

Dnsspoof

Arpspoof must be running to ensure that the DNS request 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.

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 pinging the attacking machine.

Webmitm

At this point it is obvious how you could write man in the middle (MITM) tools. The Dsniff package includes several MITM tools. I played with webmitm with mixed results. Webmitm helps 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. But after the first page, the SSL interactions seem to stall out. I guess that the SSL version in webmitm (built in 2001 or 2002) does not match the versions that are currently being used.

Dsniff

Dsniff watches for passwords set in the clear over a number of different protocols. I have not played with this tool. Set up some telnet or ftp servers and see if dsniff finds anything.

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 part is passing through your machine. Try out some of the following:

- filesnarf
- macof
- mailsnarf

- msgsnarf
- sshmitm
- sshow
- tcpkill
- tcpnice
- urlsnarf
- webspy