RADICL Lab 2 (Instructor's Copy)
Packet Crafting and Snort Rule Creation

Originally conducted on April 11, 13th 2005
Time Required: One class period, 50-55 min
Lab Written and Tested by: Sergio Caltagirone and David Manz

Possible Additions: have students write packets and send them against other students, write iptables firewall rules to block the packets, use other packet creation tools (e.g. nemesis and directly using the libpcap library), use another signature-based IDS, illustrate the use of anomaly-based IDS, run snort in 'inline' mode to block traffic or close connections.

[*INSTRUCTORS: This lab is the same as the student copy, except for notes written in italics and surrounded by brackets for easy differentiation.*]

[*LAB SETUP: Each machine should have "Dougs Security Distro" on a partition – networking between the machines is not necessary.*

Setup the KVM so that there is one account that only has access to the student machines, the students will login using that account. Assign the students/groups numbers ahead of time and tell them the specific machine to use from the KVM selection screen. It would be helpful to also set these machines, the server and the student boxes, into a separate VLAN to protect any other experiment running.]

[*Some of the contents of this lab were borrowed from "Packet Crafting for Firewall and IDS Audits," http://www.securityfocus.com/infocus/1791*]

Learning Objectives: To provide a practical introduction to packet crafting and intrusion detection systems (IDS), and hands-on experience with low-level packet behavior and IDS auditing.

[*While this lab is very introductory in its material, it is important that the students understand that these skills will be useful later – especially with regard to IDS and firewall auditing*]

Scenario: You are a security professional, tasked with the responsibility of testing snort's ability to detect certain types of packets for a company considering deploying snort as their IDS.

[*The scenario attempts to make the material relevant to actual activities taking place in organizations.*]

Deliverables: Write a three-page report on your activities. The report will be divided into three sections. The first (1) section will be a decomposition and explanation of the

packets crafted and the snort rules you wrote. Additionally, address any questions that are in the body of the lab. The second (2) section will describe how successful your attempt to catch your packets were as well as how (and where) to apply these skills to situations outside a laboratory environment. Keep enough notes during the experiment to create the report, as you will not be able to print or email results out of the lab. The third (3) section will involve short answers to the following questions:

1) What are some uses of hping2 (from both an attacker and defender perspective)?
2) What is a 'loose source record route' packet (lssr)? Why is it helpful and harmful?
3) What is an ECN/CWR stealth packet? How do attackers use it?
4) What uses port 3333? Is there any reason to block traffic on the port?

[*Things to look for in the reports:*

In section one (1) a clear breakdown (decomposition) of the packets and snort rules with explanations. Additionally, look for the answers to the questions that are asked throughout the lab. These questions help focus the student's attention through the lab on important points and help to reinforce new and reviewed concepts.

In section two (2), look for some discussion on if they were or were not successful in their attempts. Also, look for examples the students provide on how and where packet and rule creation would be used outside of a laboratory environment (e.g. firewall and IDS auditing, to detect novel attacks where signatures are not available, etc.)

Section three (3) is important because it allows the students to research some of the tools that they used in the lab. Every effort is made to make the lab as practical as possible, therefore the authors' utilized real attacks – these questions help the students find out about those attacks and the defenses that they created to stop them. Here are the answers to the questions:

(1) Test firewall rules, advanced port scanning, test performance of different protocols/fragmentation, path MTU discovery, transferring files through difficult firewall rules, remote OS fingerprinting, TCP/IP stack auditing. (http://www.hping.org/manpage.html)

(2) A 'loose source record route' (lssr) is an IP packet that has options 3 & 7 set. This means that the packet can specify one or more hops through the network (determine its own route). However, intervening routers are required record that the packet was successfully passed on – allowing the network to be probed. It is helpful because it allows an administrator to probe and test their own network, but attackers can use it to gain information. (http://www.gweep.net/~crimson/networks/lsrr.html) (http://www.informatik.uni-trier.de/~smith/networks/lsrr.html)

(3) When both the Explicit Congestion Notification (ECN) and Congestion Window Reduced (CWR) flags are set, it is called an ECN-Setup SYN packet (RFC 3168). This allows an ECN enabled destination to know that it is talking with an ECN enabled source. This is used in stealthy or 'half-open' OS fingerprinting: a SYN packet with the ECN and CWR flags are sent -> destination responds -> then a

RST packet is sent. By the way that the destination responds, their TCP stack can be fingerprinted since different OSes implement the TCP stack differently. Some systems do not log a completed connection; therefore, since this connection was not completed, then the scan and fingerprint may not be logged – hence 'stealthy'.
(http://www.daemon.be/~maarten/Maarten_Vanhorenbeeck_GCIA.pdf)

(4) *Port 3333 can be used by any application, however some applications that use it: the Daodan backdoor Trojan, many IRC bots listen to port 3333, Windows security uses 3333 for distributed firewalls, and many others. If your organization has no legitimate use of 3333, then any unused port (i.e. 3333) should always be blocked.*]

Tools To Be Used:

- Gentoo: A Linux distribution
- hping[1]: A packet crafting utility
- Snort[2]: An open-source intrusion detection system
- Ethereal: A packet sniffing utility
- iptables: A linux kernel-level firewall

[*nemesis can be used as an alternative to hping. hping was selected because of its prevalence and support by the general security community – however, any packet creation tool can be used*]

Directions:

You will not be able to print, email, or otherwise transport data out of the RADICL lab, so it is imperative that you take the notes necessary to compose the report.

[*As of the writing of this lab, there was not network printer, but that may have changed in the future allowing the students to print out some of their work*]

You will be using Gentoo, a Linux distribution during this lab that has been installed with a graphical manager called Fluxbox. The distribution has been bundled with a number of security tools that you are free to explore during this lab. Right click on the screen to bring up a menu and select your tool. Some tools, however, will only be available in the command line (xterm selection on the fluxbox interface).

[*If the KVM has not been logged into, tell the students about the KVM username to use and their machine to choose.]*

If your workstation has not been logged into, login to your workstation using the username 'root' and password 'password'.

[1] http://www.hping.org/
[2] http://www.snort.org/

First and foremost familiarize yourself with your machine. Determine your host IP, as you remember from the first lab use `ifconfig`. In this experiment, your machine is not connected to the larger network and so we must use the internal loopback (lo) controller for communication.

Start up ethereal from fluxbox, check that "Update list of packets in real time" is selected, and that interface is set to "any".

Next we will introduce you to the hping command. A simple command line utility, hping allows you to actually craft packets and customize the entire packet. You can run "`man hping2`" for further details. Just to familiarize yourself with hping, let's send a packet to your self by using your <IP ADDRESS> discovered with `ifconfig`. You will use the -S flag to send a SYN packet, to port 23. And you will only send one packet with the -c flag. **Try running the command below:**

```
# hping -S <IP ADDRESS> -p 23 -c 1
#
```

If you are observant you will notice a spelling error in the hping output. **Was the packet successfully sent and received?**

[*The output of the command should be something like this:*
```
--- <IP ADDRESS> hping statistic ---
1 packets tramitted, 1 packets received, 0% packet loss
```
round-trip min/avg/max = 0.5/0.5/0.5 ms

The 1 packet transmitted, 1 packet received, shows that the packet was successful]

Now setup iptables (a linux kernel-level firewall) to block port 23 (note there are two dashes before destination). **Run this command:**

```
# iptables -A INPUT -s <IP ADDRESS> -p tcp --destination-port :23 -j DROP
#
```

[*This provides a short introduction to iptables and firewall rules.*]

The flags are as follows: -s defines our IP Address, -p is the protocol, and -j defines the action to take when a packet is matched. **Now run the exact same hping command**, and you will find that your firewall is working, and the packet is blocked. **How can you tell the packet was dropped?**

[*The output of the hping command should be something like this:*
```
--- <IP ADDRESS> hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
```
round-trip min/avg/max = 0.5/0.5/0.5 ms

The 1 packet was transmitted, but not received – shows packet was dropped at firewall]

Next we can describe an ICMP echo request (thats a one not an L). **Run this command:**

```
# hping -1 <IP ADDRESS> -c 2
#
```

Notice that we are sending 2 packets in this command. **What makes this packet different from the first one sent?** Using ethereal, **how many packets are transmitted because of this command – why are more than 2 packets transmitted?**

[*An ICMP echo request (ping) should send 1 packet, and receive 1 packet – therefore, in this case, 4 packets will be seen (2 requests and 2 replies). The difference between this packet and the previous packet is that (1) it is an ICMP packet (versus a TCP packet), and (2) no port specification is necessary.*

Our last crafted packet will be a UDP probe on port 53. **Run this command:**

```
# hping -2 <IP ADDRESS> -p 53 -c 1
#
```

You can try several variations on the above themes and craft different packets using hping, but as you can tell from above, the -1 flag is icmp and the -2 is udp. Now **look at the man pages, what other types of packets is hping2 able to send?**

[*hping2 is able to control ICMP, TCP, UDP, and RAW IP*]

Snort can run in four basic modes. First is the Sniffer mode, then Packet Logger mode, next Network Intrusion Detection System mode, and last Inline mode. The Sniffer mode functions exactly like ethereal and tcpdump as you will see shortly. The Packet Logger simply takes that same data and saves it to a file. The NIDS mode allows snort to match traffic with its rule set and perform alert actions. Lastly Inline mode gets the packets after iptables (linux firewall) and then can instruct iptables to either drop or allow the traffic (we will using inline mode during this experiement).

With that introduction let's see how basic snort is similar to ethereal or tcpdump. Start snort to run in sniffer mode by **running this command:**

```
# snort -i any -v
#
```

Just like ethereal, we need to specify not only eth0 but any and all interfaces (because we are pinging and testing ourselves). The -v flag simply logs data. Now, open up a new terminal and **rerun any two of the hping commands**, you will see the live data gathering that snort performs. **Examine your output and see how it is similar, and different from ethereal**. Just like tcpdump its not easly to quickly make sense of the data. When you are done, hit ctrl-c to stop the snort process.

[*Ethereal and snort show the same packets, except that snort shows all of the characteristics of the packet, whereas ethereal separates the characteristics between multiple windows for GUI purposes.*

The purpose of having the students see the difference is that they may have to use snort, tcpdump or similar application in the future, and should know to recognize such a format.]

Next we can move along to packet logger. Clearly it will gather the same data as before, but now it will save it to disk. So let's go ahead and **make a log directory now by running** "`mkdir log`". **Run this new snort command**:

```
# snort -i any -dev -l ./log
#
```

Rerun your suite of hping commands. After you have run several hping commands, exit out of snort using ctrl-c, and navigate to the log directory. You will note that the default file structure is to categorize the packets by their source IP. Within each directory snort organizes the files by communication type. Browse around and familiarize yourself with the same information as snort sniffer, tcpdump, and ethereal, this time the data is logged on your hard drive and searchable. This becomes very useful when trying to automate snort.

[*This step was included because snort's logging feature is one of its most valuable for automation and system auditing. The students should be aware of this feature as it is common.*]

Now we finally get into the IDS portion of snort. We could still specify a location to log all this info, but if we do not specify a directory (as above), it will default to /var/log/snort, which is fine. We start snort in IDS mode by **running the below command:**

```
# snort -i any -A full -c /etc/snort/snort.conf
#
```

When you run that command sort will use the default rule set. Again craft several different packets (as above) and notice how snort logs them in the /var/log/snort directory. Now notice that unlike the sniffer and loggers, we have an alert file. So let's navigate there at see what it displays when we **run the command below to create a 'loose source record route packet' (lssr)**:

```
# hping -S --lsrr <IP ADDRESS> <IP ADDRESS> -p 25 -c 1
#
```

[*This command generates an output like:*

```
--- 1 <IP ADDRESS> hping statistic ---
1 packets tramitted, 0 packets received, 100% packet loss
```

round-trip min/avg/max = 0.0/0.0/0.0 ms

The 100% packet loss shows that the packet was not received.]

As you can see the packet was NOT received. So let's go back to the alert file and see what it says. **Find the alert that snort generated, what differentiated that alert from the others?**

[*The alert that snort generates is:*

```
[**] [1:501:2] MISC source route lssre [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
03/03-09:22:33.600331 0:C:6E:8C:D4:61 -> 0:50:DA:C5:9D:8B type:0x800
len:0x3E
192.168.1.100:1636 -> 192.168.1.108:25 TCP TTL:64 TOS:0x0 ID:57275
IpLen:28
DgmLen:48
IP Options (1) => LSRR
******S* Seq: 0x16EF6435  Ack: 0x3AC297BE  Win: 0x200  TcpLen: 20
[Xref => http://www.whitehats.com/info/IDS420
][Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0909
][Xref => http://www.securityfocus.com/bid/646]
```

Snort clearly alerts on the lssr packet, the title of the alert makes this one unique from the other BAD-TRAFFIC alerts.]

Lets **delete the alert file by running** "rm alert" **and restarting snort**, so that we remove the previous alerts. Now try crafting a packet with the Explicit Congestion Notification (ECN) and Congestion Window Reduced (CWR) flags set in the 13th byte offset in a TCP packet (this is a common characteristic of a stealth attack) by **running the command below:**

```
# hping -X -Y <IP ADDRESS> -p 80 -c 1
#
```

Why is snort alerting BAD-TRAFFIC? What about the packets does snort not like? Did snort differentiate between BAD-TRAFFIC and a potential stealth attack?

[*Snort is alerting BAD-TRAFFIC because the source and destination addresses of the packets are the same; also, snort does not like TCP traffic coming through over the loopback (lo) interface.*

Snort DID NOT differentiate between the ECN/CWR stealth attack and BAD-TRAFFIC.]

Let's go about writing a snort rule to catch this stealth attack. The snort configuration file uses numerous rules files but the first (and highest precedence) is /etc/snort/local.rules. **Use your favorite text editor (e.g. pico, vi) to edit local.rules**.

[*local.rules should be empty at this point*]

Let us start with a trivial case first. We are going to designate all traffic on port 3333 as suspect. So let's log an alert anytime a tcp connection is made to port 3333. **Write this rule <u>on one line</u> of local.rules:**

```
alert tcp any any -> <IP ADDRESS>/32 3333 (msg:"Contact on banned port
3333";)
```

[*This snort rule breakdown (in order):*
Action: alert, Protocol: tcp, Source IP: any, Source port: any -> Destination IP: <IP Address>, Destination Port: 3333, Set message to be: "Contact on banned port 3333";

Alerts on any tcp packet incoming to our machine on port 3333]

Anytime you add a rule to snort, you need to restart the snort command to allow the new rules to be applied. **Restart snort.**

Now let's craft a packet using hping to connect to port 3333, **run this command:**

```
# hping -S <IP ADDRESS> -p 3333 -c 1
#
```

Look at the alert file, did your snort rule alert on the packet?

[*Snort should have alerted on the packet, the alert should look like this:*
[**] [1:0:0] Contact on banned port 3333 [**]
]

We turn our attention back to our original 13 byte-offset packet with ECN and CWR flags set. It would be nice if we could catch this, because it often signifies a stealth attack. Now **add the following rule to the local.rules file:**

```
alert tcp any any -> <IP ADDRESS>/32 any (msg:"STEALTH ACTIVITY
(unknown) detection"; flags:12;)
```

[*local.rules will now contain two rules, this one and the previous one that alerts on port 3333 traffic*
Snort rule breakdown (in order):
Action: alert, Protocol: tcp, Source IP: any, Source port: any -> Destination IP: <IP Address>, Destination port: any, Set message: "STEALTH ACTIVITY (unknown) detection", Look if TCP flags 1(ECN) & 2(CWR) are set;

This rule alerts on any TCP packet coming to our machine with flags 1 (ECN) & 2 (CWR) set.]

Restart snort and resend the ECN/CWR packet from earlier.

Check the alert file, did snort catch it this time? Why did snort catch it? In your report, decompose the snort rule above and explain the different parts, giving attention to what about the rule caught the packet.

[*Snort should have caught this packet. It alerted to the packet because the two flags were set simultaneously*.]

ADVANCED MATERIAL (if time permits):

[*This section added because it is a nice little exercise that dovetails well with the previous work and the understanding that has built up on the BAD-TRAFFIC alerts and allows them to write a packet that circumvents a snort rule – a good little trick to know from an attacker's perspective.*]

Using "man hping2", determine how to spoof the source and destination address of a packet. Clear the snort alerts and restart snort. Send a packet, using the spoof abilities of hping2, such that snort does not give a BAD-TRAFFIC alert.

[*To remove the BAD-TRAFFIC alert, the source and destination address of the packet must be different. The hping command to do this is:*

```
hping x.x.x.x --rand-dest --rand-source --interface <INTERFACE> –c 1

or

hping –a <SOME SRC IP ADDRESS> <DEST IP ADDRESS> -c 1

By default, in hping, the last IP Address given is the destination (so
no switch is required to set a destination IP).
```

The interface in our experiment was the loopback interface (lo) although it may be different depending on the network setup (e.g. eth0).]