RADICL Lab 2
Packet Crafting and Snort Rule Creation

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.

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.

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?

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

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.

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).

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

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

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?**

Now let's quickly setup your 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
#
```

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?**

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?**

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?**

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.

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.

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
#
```

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?**

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?

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**.

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 on one line of local.rules:**

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

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;)
```

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.

ADVANCED MATERIAL (if time permits):

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.