

CS 491S: Computer and Network Security

Fall 2008

Lab exercise: Working with Wireshark and Snort for Intrusion Detection

Abstract:

This lab is intended to give you experience with two key tools used by information security staff. Wireshark (once Ethereal), originally written by Gerald Combs, is among the most used freely available packet analysis tools. The second is the Snort program written by Marty Roesch and a host of contributors. Snort is a simple and powerful network-monitoring agent. You will use wireshark to capture traffic destined specifically for your host, and then use snort to analyze the packet trace.

1 Tools required for this lab:

- Access to a machine with wireshark and snort installed. You are best to do this on a personal machine, or even on a VM.
- A wired network jack.

2 Pre-lab Background:

The suggested background reading may help you complete the questions..

- The wireshark homepage <http://www.wireshark.org/>

Specifically the FAQ and the Documentation links

<http://www.wireshark.org/faq.html>
<http://www.wireshark.org/docs/>

- The snort homepage. www.snort.org. On the homepage there are a few documents that may assist you in understanding snort:

Snort FAQ

<http://www.snort.org/docs/FAQ.txt>

Snort Overview

http://www.snort.org/docs/snort_manual/node2.html

How to Write Snort Rules and Keep Your Sanity

http://www.snort.org/docs/snort_manual/node195.html

The writing snort rules document is an especially helpful reference for writing the snort rules needed for this lab.

3 Lab exercises: Wireshark

3.1 *Please complete the following exercises. As always, you must hand in a lab write up containing answers to questions asked for each task.*

If wireshark is not installed already, you may need to install `libpcap` first. Read the documentation. You may be able to find these as a package, or may have to build each from source. You should be able to figure out how to install it yourself, If not google will likely get you to a suitable answer.

If snort is not installed already, install `libpcap` and then `snort`. You may be able to find these as a package, or may have to build each from Source. You should be able to figure out how to install snort yourself. If not google will likely get you to a suitable answer.

You can run this lab on your own system since it is a lab for setting up defenses --- but don't be stupid: **wireshark and snort are easily viewed as a packet-sniffing tool and you may be accused of hacking**. For this lab, we will be sniffing live packets and reading packet traces from a file. Make **certain** that you are only looking at traffic destined for your specific host. I wouldn't risk it if you have **any** concerns about how your activities may be viewed by others.

Snort, like wireshark can behave similar to `tcpdump`, but has cleaner output and a more versatile rule language. Just like `tcpdump`, each will listen to a particular interface, or read a packet trace from a file.

First we need to generate a packet trace that we will then analyze with wireshark and write snort rules for. In this lab you will select several websites to connect to.

Before we run wireshark, you need to determine how best to write a `packetfilter` that will collect only the traffic from your host. If you are on a switched network, this will be the default mode of operation. *You do **NOT NEED TO** to sniff any traffic but that destined for your host.* If you use ARP spoofing, TCP hijacking, bridge table overflows, etc, you will be trying to bypass security controls on the network. We are not asking you to do anything that will be viewed as violating security controls.

See: <http://www.wireshark.org/faq.html#promiscsniff>

However, this is a lab exercise, and we want you to practice writing filters, even if you are on a switched network. Review the following link to write up an appropriate capture filter. See the following if you have questions about writing capture filters. This means you also need to determine the IP address of your host.

http://www.tcpdump.org/tcpdump_man.html

Select a website that is not likely to be in your DNS or browser cache. The easiest way to do this is to select a site that you do not often visit.

We need to create a tracefile. This tracefile will include possibly the ARP traces for the default gateway lookup, the DNS query and response for the hostname to IP address mapping, and the HTML content transferred over the http connection to the website in question. Select both an HTTP: URL as well as an HTTPS. URL to connect to.

Reading the documentation, you should be able to determine how to create a packet capture file containing the above content pursuant to the capture filter that you wrote. Make sure to save the packet capture to a file for later analysis.

Once you have the packet capture file, you will need to write capture filters for the following.

- Display all ARP traffic
- Display all DNS traffic to/from your host
- Display all HTTP and HTTPS traffic to/from your host

Commonly security administrators are asked to look at a packet trace to analyze a recent attack. In this lab, we are going to learn how to use snort to read traces and learn how to write new snort rules.

You can always get a list of command line options by typing "snort -help". A good set of command line arguments to pass snort in this lab is:

```
snort -r /tmp/snort-ids-lab.log -P 5000 -c /tmp/rules -e -X -v
```

The intention of snort is to alert the administrator when any rules match an incoming packet. Administrators can keep a large list of rules in a file, much like a firewall rule set may be kept.

All the rules are generally about one line in length and follow the same format. Here's an example

```
log tcp any any -> 128.119.245.66 23 (msg: "telnet to www machine!");
```

This rule tells snort to record ("log") all packets destined to the telnet port on 128.119.245.66 and to include a user readable string.

In general, all rules are of this form:

```
action protocol address port direction address port (rule option)
```

In our example, the action was "log". We could simply write to a common alert file with the command "alert". The difference between log and alert is that each IP address gets its own log file for later analysis, while all alerts are stored in one common file.

The protocol field can be "tcp", "udp", or "icmp". "Any" is not allowed. Addresses can be specified in CIDR notation, and ports can be given as ranges and with the "!" operator. For example, the example below (stolen from the documentation!) logs all packets to a range of machines not on ports 6000-6010.

```
log tcp any any -> 192.168.1.0/24 !6000:6010
```

The direction operator is either "->" or "<-" or "<>" for bi-directional traffic between two addresses.

The rule options specify tasks to be performed if the addresses and protocols match.

Note that several options can be listed in the parentheses. Each must end with a semicolon, even if there is only one rule. Other useful options include, "content", "flags", "ipoption". More are listed in the "writing snort rules" document.

3.2 What to hand in

Question 1.

- a. Describe how you found the ip address of your host
- b. Write out the process you used for creating a capture filter including the actual filters
- c. Write out how you created display filters to look at just DNS and just http traffic.

Question 2.

DNS is fundamental to the operation of the internet. Using your packet trace, identify the entire DNS conversation. This will be easiest using the display filter you created above. Answer the following questions

- a. Which DNS server(s) do your host connect to? Note the IP address of the servers.
- b. Do you query a single or multiple DNS servers? Does your host directly communicate with the destination's DNS server? Why or why not?
- c. Is there any authentication associated with DNS responses that you can derive from the packet trace? Do some research and defend your answer citing sources as appropriate.

Question 3.

HTTP is a principal protocol for transferring data both user-directed and machine-directed. Take a look at the HTTP and HTTPS transaction in the packet capture you created. Use a display filter to limit to just this traffic.

- a. Review the MIME header messages in the HTTP transaction; what fields are included in the MIME header? Describe the headers.
- b. Review the same MIME information in the HTTPS header; How does this information compare with the data from the HTTPS connection?

Question 4. State how each of the following real rules from the snort home page work:

1. `alert tcp $HOME_NET 23 -> $EXTERNAL_NET any (msg:"TELNET login incorrect"; content:"Login incorrect"; flags: A+; reference:arachnids,127;)`
2. `alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"EXPLOIT BIND Tsig Overflow Attempt"; content:"\80 00 07 00 00 00 00 01 3F 00 01 02/bin/sh");)`
3. `alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"SCAN FIN"; flags: F; reference:arachnids,27;)`
4. `alert tcp $EXTERNAL_NET any -> $HOME_NET 23 (msg:"MISC linux rootkit attempt lrkr0x"; flags: A+; content:"lrkr0x");)`
5. `alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-CGI view-source access "; flags: A+; content:"/view-source?../../../../../../../../etc/passwd"; nocase; reference:cve,CVE-1999-0174;)`

7. *alert icmp any any -> any any (msg:"ICMP Source Quench"; itype: 4; icode: 0;)*
8. *alert tcp \$EXTERNAL_NET any -> \$HOME_NET 53 (msg:"DNS EXPLOIT named overflow"; flags: A+; content:"thisissometempspaceforthesockinaddrinyeahyeahiknowthisislamebutanyway whocareshorizontitworkingsoalliscool"; reference:cve,CVE-1999-0833;)*
9. *alert tcp \$EXTERNAL_NET any -> \$HOME_NET 139 (msg:"NETBIOS SMB ADMIN\$access"; flow:to_server,established; content:"\\ADMIN\$\00 41 3a 00"; reference:arachnids,340; classtype:attempted-admin; sid:532; rev:4;)*
10. *alert ip \$EXTERNAL_NET \$SHELLCODE_PORTS -> \$HOME_NET any (msg:"SHELLCODE sparc NOOP"; content:"\a61c c013 a61c c013 a61c c013 a61c c013"; reference:arachnids,355; classtype:shellcode-detect; sid:646; rev:4;)*

Question 5, Develop your own snort signature to capture DNS queries directed against the host the you choose to connect to via HTTPS. Make sure that your snort rule references the DNS data and not simply IP address of the server. Include your rule as part of the answer to this question. Run the packet trace you created against the snort signature to ensure that it matches.

3.3 Gimmiv.A Analysis

Read the analysis at the below links:

<http://www.microsoft.com/technet/security/Bulletin/MS08-067.msp>

<http://blog.threatexpert.com/2008/10/gimmiva-exploits-zero-day-vulnerability.html>

Question 6. The threat expert link above describes Gimmiv.a as:

“it could technically be classified as a network-aware trojan that employs functionality of a typical RPC DCOM network-aware worm to attack other hosts in the network.”

Describe in your own terms what the above quote means. Focus on the behavior and explain how the code could impact a network. You will likely have to do some research to explain this sufficiently.

Question 7. Do some additional research about the RPC/DCOM worms. There is a long history of RPC worms that have occurred over the past several years. Research the defensive techniques that have been used to mitigate and control previous RPC worms. Describe a series to techniques that would be sufficient to mitigate the risk of Gimmiv.a

This question is intentionally vague and is designed to have you apply some of the skills you have acquired to tackling a real world problem such as being able to detect malicious activity even when you do not have access to the content of the data streams. Explain in a few paragraphs what other tools and techniques you may use to detect this signature. Provide enough detail so that a campus network administrator could follow your explanation to deploy your system in production.

Question 8. What techniques would you use to minimize the number of false positives with your technique described above.

4 Evaluation

Question 9: How hard was this lab? Was it fair? How would you change it to improve it?