E0256: Homework 2

Please see class schedule for due dates

This homework is to be done individually. You can discuss homework problems with your friends, but the writeup and code must be your own. Files containing solutions are expected to be sent by email to the instructor.

The goal of this homework is to get you familiar with some commonly-used tools in network security, such packet analysis tools and libraries. It also familiarizes you with some Web security attacks. This homework is based in part on Stanford CS155 taught by Dan Boneh and UW-Madison CS642 taught by Thomas Ristenpart.

The following deliverables are expected:

- A PDF file containing solutions to Q1 and Q2.
- A single `.c` or `.cpp` file with your code for Q3. Also expected is a `Makefile` for this program. We should be able to compile your program by just typing `make` on the command line, and get the executable program.

For Q1, Q2 and Q3, you will be working with some tools and libraries commonly used by network security analysis. You will be using the virtual machine images from Homework 1. You will first install the required tools on these virtual machines, and then use them for the homework.

To install the required tools in the virtual machines, first log in as `root` into the virtual machine from Homework 1. Then execute the following commands (in this order)

- `apt-get install libpcap-dev`
- `apt-get install tcpdump`

Q1, Q2, and Q3 also require you to access two files, `traces.tar.gz` and `scanner.c`, which are available from the following website: `http://www.csa.iisc.ac.in/~vg/teaching/E0-256/hw2/`

Question 1 — Network Packet Analysis

As part of this homework, you are given a `tar.gz` file containing packet traces in pcap format. For this question, you will learn how to read and analyze such packet capture traces. You are free to use any open-source tool for this analysis, such as `tcpdump`, which you just installed on your VMs. `tcpdump` produces its output in textual format, and has a number of flags that you can use to configure how the output looks. Use `man tcpdump` to learn about the flags that `tcpdump` uses.

You are not bound to use `tcpdump` for this question. A number of other packet scanning tools, such as WireShark, offer GUI-based interfaces for packet trace analysis. You are free to download, install and use such tools. However, if you use any of these tools, we won't be able to answer questions about how to install or use them.

If you use `tcpdump`, note that it is a tool that is often used to dump the traffic on a network. However, it can also be used to analyze packet dumps in `.pcap` format, and you will be using it in that mode. For this problem there are four `.pcap` traces in `traces.tar.gz` that you will need to investigate in order to find out the information asked for below.

Part (a) Trace 1: HTTP traffic.

1. Give three websites visited from source IP address "192.168.0.100"
2. Give three search queries made from source IP address "192.168.0.100"

Part (b) Trace 2: FTP traffic. FTP is the file transport protocol. There is a lot of information about it on the Internet, which you should read up to answer the questions below.

1. What is the user name and password used to connect to the FTP server?
2. Explain the difference between a passive FTP connection and an active FTP connection.
3. Give the packet number ranges across which there were active connection(s).
4. Give the packet number ranges across which there were passive connection(s).
5. List the names of any files that were downloaded.

Part (c) Trace 3: Traceroute. Traceroute is a tool used to determine the route between two IP addresses.

1. Identify the source IP address that issued a traceroute command.
2. Identify the destination IP address of the traceroute command.
3. List the IP addresses on the route between the source and destination.

Part (d) Trace 4: POP. The POP protocol is used for Email.

1. What is the POP username and password?
2. How many emails are in the user's mailbox?
3. Give the contents of from, to, subject, and date for each email.

(2 points for each of the 13 sub-questions, and 4 points for learning to use a packet trace analysis tool such as tcpdump).

Question 2 — Network-based Denial-of-Service

In a SYN denial of service (DoS) attack, an attacker sends a large number of SYN packets (TCP/IP packet with SYN flag set) to a victim with a spoofed source IP address. Suppose a well-intentioned, but naive network engineer decided to setup a system with a custom IDS program that sniffs traffic using the `pcap` library (to capture network packets) and logs all the TCP SYN packets to a file for later inspection. A testing version of the C code implementing the scanner is given in the file `scanner.c`. This particular implementation reads packets from a `pcap` file in offline mode, but the code that network engineer writes will replace the call to `pcap_open_offline()` with the logic needed to use `pcap_open_live()` (read live packets and store to a file).[1] The engineer has asked you for your opinion about his proposal.

[1] See `http://yuba.stanford.edu/~casado/pcap/section1.html` and `http://www.tcpdump.org/pcap.html` for good overviews of the `libpcap` library.

Part (a) List as many ways an attacker can abuse such a setup as you can think of.

Part (b) For each issue, explain how you would address it.

Question 3 — A Simple NIDS

In this problem, you will learn about three kinds of network attacks, and write a simple intrusion detection system to detect potential attacks or dangerous behavior in network activity. First, let us define the attacks. The Internet has plenty of information about these attacks, and you are encouraged to look up sources on the Web.

- ARP spoofing. The Address Resolution Protocol (ARP) is used to bind the MAC address of a host to an IP address. In an ARP spoofing attack, an attacker sends fake ARP messages to bind the attacker's MAC address with the IP address of another host.
- TCP SYN port scan. The TCP SYN message is the first message sent as part of a TCP handshake, during connection establishment. A TCP SYN port scan refers to an attack whereby a malicious attacker sends TCP SYN messages to various ports to see which ports have services listening on them. If there is an active service on a port, it will respond with a TCP SYN-ACK packet.
- TCP SYN flood. In such an attack, the goal of an attacker is to send a large number of TCP SYN messages to a victim. The victim responds with a SYN-ACK, but the attacker never responds. The hope of the attacker is that by getting the victim to initiate a large number of half-open connections, the victim will exhaust its resources, causing denial-of-service.

In the `traces.tar.gz` file, there are three pcaps with example attacks:

- `arpspoofing.pcap` includes an ARP spoof attack. IP address 192.168.0.100 advertises the wrong MAC address for 192.168.0.1.
- `portscan.pcap` includes a TCP SYN port scan.
- `tcpflood.pcap` includes a TCP SYN flood.

Your job is to write a software IDS executable (in C/C++) that takes as input a pcap trace and looks for such malicious behavior. The local network you are protecting is configured with two machines (192.168.0.100 with MAC address 7c:d1:c3:94:9e:b8 and 192.168.0.103 with MAC address d8:96:95:01:a5:c9) and a router (192.168.0.1 with MAC address f8:1a:67:cd:57:6e).

Your scanner should:

Part (a) Detect ARP spoofing attempts. Output a warning including the offending MAC address and the packet number of the offending packet.

Part (b) Detect port scans. A port scan is defined to occur whenever TCP SYNs or UDP packets are sent to a 100 or more different ports on a target system (this is a somewhat arbitrary threshold set for the purposes of this homework. In the real-world, quite a bit of tuning is necessary to find the right threshold). The scanner should output a warning including the offending source IP address, the victim destination IP address, and the offending packet numbers.

Part (c) Detect TCP SYN floods. Your tool should detect when the number of TCP SYNs to a particular destination (that are not associated with completed handshakes) exceeds 100 per second. The scanner should output a warning including the offending source IP address, victim destination IP address, and the offending packet numbers.

Your program should take as input the filename of a pcap file that contains captured network packets. The output of your program will be the warning messages as described above. The format of your result is free but it should be clear to the user. You should use C/C++. Both have libraries for reading, parsing, and analyzing pcap files — the `libpcap-dev` library that you installed is the most popular one.[2]. Your turned-in files should include a Makefile that properly compiles it. Check that your scanner runs properly on your VMs before turning it in. We will assume that you are just using the `libpcap-dev` library for your code. If you use other libraries, please send us instructions (via your homework submission) on how to install it in the VM. Remember, if we cannot compile or build your program, you will not receive credit for the question.

The sample pcap files can be used to test your scanner. We may test your scanner on fresh pcaps we generate that include other non-malicious behaviors, as well as boundary conditions (e.g., a TCP SYN flood that does not exceed 100 packets per second).

Our grading policy will be as follows:

- If we cannot build your program by just typing `make` on the command line, you will receive 0 points. If use any special libraries that are not already installed in the VMs, please send us clear instructions on how to install the library.
- If your program builds, but crashes when fed one of the three pcap files above, you will receive 0 points.
- If your program builds, runs successfully on the pcap files provided, but provides the wrong answers, you will receive 20% of the credit for that pcap file (e.g., 2 points for running on the ARP spoofing trace, but not producing the correct output).
- If your program builds, runs successfully and correctly on the pcap files provided, but runs incorrectly or crashes on a pcap file that we use to test your program, you will receive some partial credit.

[2]See `http://yuba.stanford.edu/~casado/pcap/section1.html` and `http://www.tcpdump.org/pcap.html` for a good overview of the pcap library