Project 4: Network Security

This project is due on **Saturday, December 7, 2019** at **10 PM.**. **You should work in teams of two** and submit one project per team. You will have a budget of five late days (24-hour periods) over the course of the semester that you can use to turn assignments in late without penalty and without needing to ask for an extension. You may use a maximum of two late days per assignment. Late pair projects will be charged to both partners. Once your late days are used up, extensions will only be granted in extraordinary circumstances.

The code and other answers your group submits must be entirely your own work, and you must adhere to the Code of Academic Integrity. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with anyone outside your group. You may consult published references, provided that you appropriately cite them (e.g., with program comments), as you would in an academic paper.

Solutions must be submitted electronically **via Canvas**, following the submission checklist at the end of this file.

Introduction

This project will introduce you to common network protocols, the basics behind analyzing network traces from both offensive and defensive perspectives, and local network attacks.

Objectives

- Gain exposure to core network protocols and concepts.
- Understand offensive techniques used to attack local network traffic.
- Learn to apply manual and automated traffic analysis to detect security problems.

IMPORTANT! Read This First

This project asks you to perform attacks, with our permission, against a target network that we are providing for this purpose. Attempting the same kinds of attacks against other networks without authorization is prohibited by law and university policies and may result in *fines, expulsion, and jail time*. **You must not attack any network without authorization!** There are also severe legal consequences for unauthorized interception of network data under the Electronic Communications Privacy Act and other statutes. You are required to respect the privacy and property rights of others at all times.

You may discover in the course of doing this project that your hardware is incompatible with some of the software we recommend. If neither you nor your partner has compatible hardware, you may borrow an external wireless card from us. Supplies are limited, so we recommend that you determine early whether you will need to do so and plan accordingly. External wireless cards will be available during office hours.

Kali Linux (https://www.kali.org/) has many of the tools we will use pre-installed and is available as a bootable USB image. This will also be available during office hours.

You will be corresponding over email with a fictitious intelligence agency as part of this project. In most cases, you can expect responses to be quick. If you are left waiting for more than 15 minutes, or believe you have found an error not caused by you, inquire on Piazza.

Part 0. Binary analysis

Before completing the following tasks, you *must* register you and your partner in a project group on Canvas. Please also choose an alias, which we will use to refer to your group publicly. You may use a pseudonym to keep your identity hidden if you wish. You should use your Canvas-associated email for the rest of this project.

You have received two email communiqués. Follow the instructions given in the second email. The purpose of this task is to analyze a binary file, obtain credentials to log into the administration panel, and register your group's information. For the purposes of this project, you have our permission to use the administrative credentials you obtained on the website located at https://remlink.xyz.

What to submit. As you do this project, keep a text file called `writeup.txt` with answers to the written questions. For this part, include:
(0) a brief text description of how you solved this part.

Part 1. WEP Cracking

We have set up a WiFi network encrypted with WEP, a common but insecure wireless security protocol (you can read about WEP online). You will learn the name and location of this network once you have completed the previous part. This network has been created specifically for you to attack as described below, and you have permission to do so.

First, you will need to attack the wireless network in order to find the WEP encryption key and join the network. There are many online tutorials and automated tools available to help you perform this task. We recommend Aircrack-ng, available at http://www.aircrack-ng.org/. **However, you should NOT intentionally inject any packet into the WiFi network**. Use only passive approaches.

The Aircrack website includes a helpful list of wireless cards that are compatible with their software. If neither your nor your partner's cards are compatible with this software you may borrow a USB network card from us. You only need specific network cards for this part of the assignment. We

strongly encourage you to do this part as early as possible to avoid competition for a very limited number of spare network cards. Once you know the WEP key, you can connect to the WiFi network like you would any other network (any WiFi card can do it).

What to submit Append the following to `writeup.txt`:

(1.1) the WEP key for the network;
(1.2) a brief explanation of the vulnerability you are exploiting in WEP;
(1.3) a paragraph explaining the steps and tools you used for this attack (be specific!).

Part 2. Network exploration

In this part of the project, you will analyze the network you have broken into.

Part 2.1. Data exploration

Your first step is to gather some interesting data. Take a packet capture of *all* of the traffic going through our WiFi network and save it as `dump.pcap`. You can do this using `tcpdump` or Wireshark (tcpdump is a command line tool and Wireshark has a GUI). You should read online (or in `man` pages) about these tools, and use whichever one you prefer. Make sure you save the network trace in pcap format and not pcapng format, so you can use it for the next section. You should take at least one minute of data. You will submit this as part of your report.

By default, a network interface controller receives all packets that are being sent over the network, and forwards only the frames that are being sent to your computer's Media Access Control (MAC) address; all others are dropped. In order to view all of the traffic on a network, you will need to enable promiscuous mode for your network interface, which simply requests that all frames be sent to your CPU. You can enable promiscuous mode using Wireshark or on the command line before invoking `tcpdump`.

For some OSes/hardware, promiscuous mode might not show anything besides the packets sent from/to your device. In such cases you need *monitor mode*, which captures not just the traffic sent on your specific network, but on *all* networks near your device! To enable monitor mode, your device must support it. Check tools like `airmon-ng` to set your device into monitor mode. You can then have Wireshark listen on the resulting monitor device. Note that you will want to set Wireshark or your packet capturing software to decrypt the raw 802.11 traffic with the network's WEP key. Also, if you are seeing very few packets, you are likely not listening on the right channel. You will have to figure out what the right channel is and use `airmon-ng` to set your device to that channel.

Examine your pcap file using Wireshark. Explore the different options for filtering and reconstructing data streams. This will help you understand the protocols involved.

Part 2.2. Packet analysis

In Part 2.1, you manually explored a network trace. Now, you will programmatically analyze a pcap file to detect suspicious behavior. Specifically, you will be attempting to identify port scanning.

Port scanning is a technique used to find network hosts that have services listening on one or more target ports. It can be used offensively to locate vulnerable systems in preparation for an attack, or defensively for research or network administration. In one kind of port scan technique, known as a `SYN` scan, the scanner sends TCP `SYN` packets (the first packet in the TCP handshake) and watches for hosts that respond with `SYN+ACK` packets (the second handshake step).

Since most hosts are not prepared to receive connections on any given port, typically, during a port scan, a much smaller number of hosts will respond with `SYN+ACK` packets than originally received `SYN` packets. By observing this effect in a packet trace, you can identify source addresses that may be attempting a port scan.

You will develop a Python 2 program that analyzes a pcap file to detect possible `SYN` scans, using the `dpkt` packet manipulation library. `dpkt` is available at https://github.com/kbandla/dpkt. You can view the documentation at https://dpkt.readthedocs.org/en/latest/. You can also find a helpful tutorial here: http://jon.oberheide.org/blog/2008/10/15/dpkt-tutorial-2-parsing-a-pcap-file/.

Your program will take the path of the PCAP file as a command-line parameter:

```
  python analyze.py <pcap file>
```

The output should be the set of IP addresses (one per line) that sent more than 3 times as many `SYN` packets as the number of `SYN+ACK` packets they received. Your program should silently ignore packets that are malformed or that are not using Ethernet, IP, and TCP.

A sample pcap file captured from a real network can be downloaded at ftp://ftp.bro-ids.org/enterprise-traces/hdr-traces05/lbl-internal.20041004-1305.port002.dump.anon. (You can examine the packets manually by opening this file in Wireshark.) For this input, your program's output should be these lines, in any order:

```
128.3.23.2
128.3.23.5
128.3.23.117
128.3.23.158
128.3.164.248
128.3.164.249
```

Part 2.3. Scanning

`nmap` is a network scanning tool which uses SYN scanning and other techniques to discover hosts on a network, find ports open on a particular machine, and identify software running on those machines. The basic uses of nmap will suffice for this project. Note that running nmap as root may give different responses from running it as a normal user.

Install nmap (available from http://nmap.org/ or your favorite package manager), then experiment! What happens when you `nmap localhost` on your own computer? On eniac? What happens when you run an application that listens on a port (e.g., `nc -l 31337`) and try again?

Use nmap to scan the local network you gained access to in the previous part for all machines that are alive, then check each machine individually to find out what ports they have open. You will submit the output of your nmap queries as part of your writeup.

What to submit

- `dump.pcap`: the packet capture file.
- `analyze.py`: a Python 2 program that accomplishes the packet analysis task specified above. You should assume that `dpkt` 1.9 is available, and you may use standard Python libraries, but your program should otherwise be self-contained. We will grade your analyzer using a variety of inputs.
- `nmap.txt`: the results of your nmap interrogations of our local network, which should show for each live host on the network which ports are open;
- Append the following to `writeup.txt`:
  (2) the maximum jail time you could face under 18 USC § 2511 for intercepting traffic on an encrypted WiFi network without permission.

Part 3. Email security

In this part of the project, you will need to send an email with forged header information. There is a mail server on our network that receives and sends mail. You should figure out its IP address and port from your results in Part 2. We suggest that you read about the SMTP protocol online, connect to the mail server, and send it the spoofed email (you might want to write a program to do this). You have our permission to falsify information sent to this mail server.

What to submit Append the following to `writeup.txt`:
(3.1) a description of how you sent the email. If you wrote code, include your code here.
(3.2) a brief discussion of countermeasures against this attack.

Part 4. Password security

For this part of the project, you will write a C program to log into a server by guessing the password. You will use a dictionary attack rather than a simple brute-force method.

The server in question runs a custom protocol which requires a username, password, and secondary authentication token, which you have acquired in the previous part of this project. We suggest first using the `telnet` program to connect to the server and reverse engineer the protocol. You may log in with the user `steve` and his password `onemorething`, but his account does not have the same authorization as the `ceo`.

Your program should take in the following command-line arguments:

```
./cracker <server ip> <port number> <username> <2fa token> <dictionary file>
```

If your program succeeds at finding the password, it should print it to standard out. If it does not, there should be no output. Either way, your program must terminate by itself. You can find some framework code to open a socket, send "Hello World!", and listen for a response at https://www.cis.upenn.edu/~cis331/project4/client.c. A corresponding Makefile is at https://www.cis.upenn.edu/~cis331/project4/Makefile.

You can try out our framework code using netcat. To listen on a port, run:
OS X: `$ nc -l 31337`
Linux: `$ nc -l -p 31337`

Then connect to the port: `$ ./client 127.0.0.1 31337`.

A word file is available at http://downloads.skullsecurity.org/passwords/john.txt.bz2.

What to submit `cracker.c, Makefile`: A C program that accomplishes the task specified above. You should only need to use the libraries given in the framework. Include a Makefile so your client can be easily rebuilt.

Append the following to `writeup.txt`:
(4) a brief discussion of countermeasures against brute-force password cracking.

Part 5. DNS hijacking

In this section, you will use a DNS management panel to hijack the command and control domain being used for the malware. Finally, you will use the socket programming skills you learned earlier to terminate all listening instances of remlinK's malicious antivirus devices.

What to submit Append the following to `writeup.txt`:
(5.1) a brief explanation of what DNS is used for and how it is misused by malware authors.
(5.2) a brief explanation of how you solved this part.
(5.3) a brief discussion of countermeasures an entity could take to prevent DNS hijacking.
(5.4) the maximum jail time you could face under 18 USC § 1030 for your actions (explain each count), if your actions were not authorized.

Part 6. Active network attacks [Extra Credit]

There is a host on the network communicating over SSL-encrypted email. To get access to the data being sent, you will conduct a man-in-the-middle (MITM) attack on the connection between the mail server and the mail client. Instead of allowing the two to directly execute the SSL handshake, you will force both the server and the client to connect securely to your computer, while believing that they are connecting directly to the other.

In order to act as a MITM, you will need to convince both the client and the server to send traffic to you. One way to do this on a local network is ARP spoofing, in which the attacker introduces false data into the client's Address Resolution Protocol (ARP) table. The ARP table gives a mapping between IP addresses and physical MAC addresses, and is populated by hosts sending out broadcast packets querying the MAC address of a specific IP address. An attacker can spoof responses to these ARP requests.

For this attack, you may want to use `arpspoof`, and `sslsniff`, but you are not limited to just these two. There are many available. Be aware that hardware support for this attack is limited. We suggest consulting the TAs and borrowing hardware sooner rather than later.

Other groups working on the assignment simultaneously may interfere with this section of the project. We have set the router for the network to reset itself every ten minutes, so if your results seem abnormal, wait a few minutes and try again.

What to submit Append the following to `writeup.txt`:
(6.1) an explanation of the vulnerability you are exploiting;
(6.2) a paragraph explaining the steps and tools you used for this attack;
(6.3) a brief discussion of countermeasures against this attack.
(6.4) a description of any other vulnerabilities you were able to exploit on the network that aren't listed elsewhere, along with any information you were able to obtain.

Submission Checklist

You have been interacting with your email correspondent throughout this project. You must complete all of the tasks your correspondent gives you to receive full credit for the project. *In addition*, you must also submit your writeup and project code to Canvas in a gzipped tarball (`.tar.gz`) named `project4.`*`pennkey1`*`.`*`pennkey2`*`.tar.gz`. The tarball should contain only the files below:

Part 2: Packet Analysis

`dump.pcap`: your PCAP data;
`analyze.py`: a Python2.x program for detecting SYN scanning, as specified in Part 2.
`nmap.txt`: the results of your nmap interrogations of our local network

Part 4: Password Security

`cracker.c, Makefile`: A C program to brute force the password for our custom protocol.

All

`writeup.txt`:

```
(0) a brief explanation of part 0;

(1.1) the WEP key;

(1.2) a brief explanation of the WEP vulnerability;

(1.3) a paragraph explaining the steps and tools you used for this attack.

(2) maximum jail time for intercepting WiFi traffic without permission;

(3.1) a description of how you sent the email.

(3.2) a brief discussion of countermeasures against this attack.

(4) a brief discussion of countermeasures against brute-force password cracking;

(5.1) a brief explanation of what DNS is used for and how it is misused

(5.2) a brief explanation of how you solved this part

(5.3) a brief discussion of countermeasures to prevent DNS hijacking
```

(5.4) the maximum jail time you could face under 18~USC~\S~1030
for your actions (explain each count), if your actions were not authorized.

(6.1) an explanation of ARP spoofing vulnerability [Extra credit];

(6.2) a paragraph explaining the steps and tools you used [Extra credit];

(6.3) a brief discussion of countermeasures against ARP spoofing [Extra credit];

(6.4) a description of any extra vulnerabilities you were able to exploit
in the network. [Extra credit]