COS432/ELE432 March 30, 2016
Information Security Project 4: Network Security

Project 4: Network Security

This project is split into two parts, with the first checkpoint due on **Monday, December 5** at **6:00 PM** and the second checkpoint due on **Wednesday, December 14** at **6:00 PM**. The first checkpoint is worth 2% of your total grade, and the second checkpoint is worth 10%. We strongly recommend that you get started early.

This is a group project; you SHOULD work in **teams of two** and if you are in teams of two, you MUST submit one project per team. If two different sets of answers are submitted, the one with lower total grade will be accepted. Please find a partner as soon as possible. If you have trouble forming a team, post on Piazza's partner search forum. Note that Aircrack-ng Suite, one of the tools required for Checkpoint 2 may not be compatible with your hardware. Please read the **Checkpoint 2 Setup** in advance. Build your teams such that at least one member of the team can run the required tools.

The code and other answers your group submits must be entirely your own work, and you are bound by the Student Code. You MAY consult with other students about the conceptualization of the project and the meaning of the questions, but you MUST 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 in any one of the group member's SVN repository, following the filename and solution formats specified under the submission checklist given at the end of each checkpoint.

"You can't defend. You can't prevent. The only thing you can do is detect and respond."
– Bruce Schneier

Introduction

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

Guidelines

- You SHOULD work in a group of 2.
- Your answers may or may not be the same as your classmates'.
- All the necessary files to start the project are given under a folder called "mp4" in your SVN repository:
- We generated files for you to submit your answers in. You MUST submit your answers in the provided files; we will only grade what's there!
- Each submission file contains an example of expected format. Failure to follow this format may result in 1 point deduction for the section. You MAY delete the examples; they will be ignored (along with any other line starting with #) when grading.

Required Tools

- Wireshark 32 bit*
- Aircrack-ng Suite
- nmap
- Python 2.7
- dpkt (Python package)

* We strongly recommend using 32 bit version of Wireshark for Checkpoint 2. Bugs within 64 bit version may prevent you from completing the task.

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. Per the course ethics policy, you are required to respect the privacy and property rights of others at all times, *or else you will fail the course.* See “Ethics, Law, and University Policies” on the course website.

4.1 Checkpoint 1 (20 points)

Security analysts and attackers both frequently study network traffic to search for vulnerabilities and to characterize network behavior. In this section, you will examine a network trace from a sample network we set up for this assignment. You will search for specific details about the network using Wireshark network packet analyzer (`https://www.wireshark.org`).

4.1.1 Exploring Network Traces (10 points)

Examine the first network trace, `4.1.1.pcap`, using Wireshark.

Provide concise answers to the following questions.

1. Multiple hosts sitting at the local network are sending packets. What are their MAC and IP addresses? (3 points)
2. How many unique TCP conversations, also known as TCP sessions, are there? (1 points)
3. The trace shows that at least one of the clients makes HTTPS connections to sites other than Facebook. Pick one of these connections and answer the following questions. Your answers should include references by number to corresponding Wireshark frames. (3 points)
   (a) What is the domain name of the site the client is connecting to?
   (b) Is there any way the HTTPS server can protect against the leak of information in (a), namely the domain name of the site the client was connecting to?
   (c) During the TLS handshake, the client provides a list of supported cipher suites. List the cipher suites and name the crypto algorithms used for each.
   (d) Based on what you have been taught and any other information you can find, are any of these cipher suites worrisome from a security or privacy perspective? Why?
   (e) What cipher suite does the server choose for the connection?
4. One of the hosts performed port scanning, a technique used to find network hosts that have services listening on one or more target ports.
   What is the IP address of the port scanner? (1 points)

Check your answer formats with the examples provided under *Checkpoint 1: Submission Checklist*.

4.1.2 HTTPS Traffic (10 points)

Examine the second network trace, `4.1.2.pcap`, using Wireshark.

Provide concise answers to the following questions.

1. In what year was this traffic captured? (1 points)
2. What is the hostname (or Fully Qualified Domain Name) of the server that the client connected to? (1 points)
3. During TLS handshakes, the client(s) provided a list of supported cipher suites.
   List the supported cipher suites from one of the clients. (1 points)
4. Which cipher suite did the server choose for the connection? (1 points)
5. A user of the client searched a person's name on the website.
   What is the first name of this person? (2 points)
6. The user sent a message to the person found in the previous question.
   What is the *body* of the message? (3 points)
7. Export and submit the user's cookie used in sending the message. (1 points)

Check your answer formats with the examples provided under *Checkpoint 1: Submission Checklist*.

Checkpoint 1: Submission Checklist

Inside your mp4 SVN repository, you will have auto-generated files named as below.
Make sure that your answers for all tasks up to this point are submitted in the following files by **Monday, December 5** at **6:00 PM**.

SVN Repository

Team Members

`partners.txt` : a text file containing NETIDs of both members, one NETID per line.
Place the NETID of the student making the submission in the first line.

example content of `partners.txt`

```
your_netid
partner_netid
```

Solution Format

example content of `4.1.1.1_ip.txt`

```
1.2.3.4
127.0.0.1
```

example content of `4.1.1.1_mac.txt`

```
0f:0f:0f:0f:0f:0f
1e:1e:1e:1e:1e:1e
```

example content of `4.1.1.2.txt`

```
461
```

example content of `4.1.1.3.txt`

```
8.8.8.8
```

example content of `4.1.1.4_active.txt`

```
content from active_FTP !@($!@:+_
```

example content of `4.1.1.4_passive.txt`

```
content from passive_FTP!@($!@:+_
```

example content of `4.1.1.5.txt`

```
192.168.1.254
```

example content of `4.1.2.1.txt`

```
1970
```

example content of `4.1.2.2.txt`

```
www.example.com
```

```
blog.example.com
```

example content of `4.1.2.3.txt`

```
TLS_ECDHE_RSA_WITH_RC4_128_SHA (0xc011)
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA (0xc012)
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
```

example content of `4.1.2.4.txt`

```
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
```

example content of `4.1.2.5.txt`

```
john
```

example content of `4.1.2.6.txt`

```
========security is kewl; blah blah blah @#$!)$(@ this message is in a si
ngle line.============================
```

example content of `4.1.2.7.txt`

```
__utma=44258684.1258198627.1456687180.1457965376.1458137415.4; __utmc=442
58684; __utmz=44258684.1457965376.3.3.utmcsr=isss.illinois.edu|utmccn=(re
ferral)|utmcmd=referral|utmcct=/
```

List of files that must be submitted for Checkpoint 1

- `partners.txt`
- `4.1.1.1_mac.txt`
- `4.1.1.1_ip.txt`
- `4.1.1.2.txt`
- `4.1.1.3.txt`
- `4.1.1.4_active.txt`
- `4.1.1.4_passive.txt`
- `4.1.1.5.txt`
- `4.1.2.1.txt`
- `4.1.2.2.txt`
- `4.1.2.3.txt`
- `4.1.2.4.txt`
- `4.1.2.5.txt`
- `4.1.2.6.txt`
- `4.1.2.7.txt`

4.2 Checkpoint 2 (100 points)

4.2.1 Network Attacks (60 points)

In this part of the project, you will experiment with network attacks by:

- cracking password for a WiFi network protected with WEP (Wired Equivalent Privacy), a security protocol designed for WiFi networks
- decrypting HTTPS connections
- recovering a simulated victim's credentials

At Siebel Center Room 1129 (CS460/ECE419 Lab), you will find a WEP-encrypted WiFi network named `cs461sp16`. We've created this network specifically for you to attack, and you have permission to do so. There is also one or more clients wirelessly connected to this network that makes connections to a HTTPS server, also wirelessly connected, every few seconds.

Your goal is to find the credentials (username and password) used to log in to the website and retrieve a secret message.

You SHOULD use suggested tools to complete the tasks. If you need to use anything else, you MUST ask a TA for permission.

If you run into issues due to having incompatible wireless adapter, you MAY rent a USB wireless adapter during TA's office hours. Rent period is 24 hours. If returning early or during weekends, please arrange a meeting with a TA by sending an email to .

Setup

- **For Macbook (OS X) Users**

  We **STRONGLY recommend** you to find a partner who has a non-Apple laptop, as you will definitely run into issues with getting the built-in wireless card work with the Aircrack-ng Suite. If you choose to use OS X to do this assignment, you have following two options, which may or may not work:

  - Use the built-in AirPort Utility on OS X
    - You can use AirPort Utility, in place of Airmon-ng and Airodump-ng (tools included in the Aircrack-ng Suite), to capture packets in monitor mode.
    - Limitation: AirPort Utility does not let you specify the wireless network you are interested in capturing. You may have to wait for hours, or even days depending on the volume of wireless traffic in the area, to capture enough packets to successfully crack the network key.
  - Use Kali Linux Live and USB wireless adapter
    - As it is difficult to find stable wireless card drivers for Linux running on Apple devices, you will have to use a USB wireless adapter that is compatible with the Aircrack-ng Suite.
    - Limitation: We have a very limited number of USB wireless adapters available for rent.

- **For Others, Use Kali Linux Live (on your own machine only)**
  We **STRONGLY recommend** using Kali Linux Live booted from a USB drive. Kali Linux (`https://www.kali.org`) is a Linux distribution designed for penetration testing. It is pre-installed with a lot of penetration-testing programs, including Wireshark, Aircrack-ng Suite, and nmap.

  - Use the most recent 32 bit version of Kali Linux:
    `http://cdimage.kali.org/kali-2016.1/kali-linux-2016.1-i386.iso`
  - To create a Kali Live USB drive, follow this tutorial:
    `http://docs.kali.org/downloading/kali-linux-live-usb-install`
  - To preserve data on the Kali Live USB drive, follow this tutorial:
    `http://docs.kali.org/downloading/kali-linux-live-usb-persistence`

  Warning: DO NOT use this drive on school computers, such as EWS or any other lab machines.

- **Do Not Use VMs**
  We **DO NOT recommend** using virtual machines. Your built-in wireless card will not work with Aircrack-ng in virtual machines. If you choose this option, we cannot help you with troubleshooting.

WEP cracking

First, you will need to crack the WEP encryption key for the network using Aircrack-ng Suite (`http://www.aircrack-ng.org`).

Aircrack-ng website provides many helpful tutorials (e.g. `http://www.aircrack-ng.org/doku.php?id=simple_wep_crack`). You SHOULD go through the tutorials to understand the purpose of each tool and learn how to use them.
As you will learn from the tutorial, WEP cracking process usually involves performing attacks to generate traffic so that data can be collected faster. For this project, you MUST NOT use those attacks as they will disrupt the network.
We've made sure that there's sufficient traffic generated on the network to allow successful WEP cracking within a reasonable amount of time (< 1 hour). If the process is taking too long, please notify a TA.
Note: You SHOULD save the network trace from this part for later use.

1. What is the WEP key for the network? (10 points)

   What to submit: Submit `4.2.1.1.txt` that contains the WEP key in *ASCII characters*.

Network analysis

Once you've cracked the WEP key, examine the network trace from the previous part using Wireshark. You will first need to decrypt the 802.11 (wireless) traffic. Wireshark can do this if you provide the correct WEP encryption key.
You may also join the network to gather more information. In addition to examining the network traffic, consider using nmap (`https://nmap.org`), a powerful tool for probing remote hosts. Determine the IP addresses of the client(s) and server and any services (below port 4096) running on these machines.
Note: Other students (and yourself) on the network are not considered as clients.

2. What are the IP addresses of the server and client(s) on the network? (6 points)

   What to submit: Submit `4.2.1.2.txt` that contains the IP addresses. Write one address per line.

3. What services does the server provide? (12 points)

   What to submit: Submit `4.2.1.3.txt` that contains application layer protocols (in standard acronyms) used by the services. Write one protocol per line.

Attacks

In order to discover the client's username and password, you will need to decrypt the HTTPS traffic. Wireshark can do this for TLS connections that do not use forward secrecy, if you provide the server's private key. Username and password will expire after 1 hour from when the packet was generated. You may also want read up on the HTTP Basic Authentication method, which is specified in RFC 2617.
Note: We strongly recommend using 32 bit version of Wireshark for this part. Bugs within 64 bit version may prevent you from completing the task.

4. What are the username and password that the client used to log in to the website?
   You may find multiple pairs; pick any one of them. (20 points)

   What to submit: Submit `4.2.1.4.txt` that contains the username in first line and password in second. You MUST submit only one pair.

5. What is the secret message displayed on the webpage after login?
   Each login attempt generates a different secret message; pick any one of them. (10 points)

   What to submit: Submit `4.2.1.5.txt` that contains the secret message. When obtaining the message, you MUST use the same credentials you submit for the previous question. You MUST submit only one.

6. What is the maximum number of years in jail that you could face under 18 USC § 2511 for intercepting traffic on an encrypted WiFi network without permission? (2 points)

   What to submit: Submit `4.2.1.6.txt` that contains the maximum number of years in jail. Write the number only.

4.2.2 Anomaly Detection (40 points)

In 4.1.1, you manually explored a network trace and detected a port scanning activity. Now, you will programmatically analyze trace data to detect such activity.

Port scanning can be used offensively to locate vulnerable systems in preparation for an attack, or defensively for research or network administration. In one port scan technique, known as a `SYN` scan, the scanner sends TCP `SYN` packets (the first step in the TCP handshake) and watches for hosts that respond with `SYN+ACK` packets (the second 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 network trace, you can identify source addresses that may be attempting a port scan.

Your task is to develop a Python program that analyzes a PCAP file in order to detect possible `SYN` scans. You MUST use `dpkt` Python library for packet manipulation and dissection.
For more information about `dpkt`:

- PyDoc documentation - `pydoc dpkt`
- official website - `https://github.com/kbandla/dpkt`

You may also find this tutorial helpful:
`https://jon.oberheide.org/blog/2008/10/15/dpkt-tutorial-2-parsing-a-pcap-file`

Specifications

- Your program MUST take one argument, the name of the PCAP file to be analyzed:
  ex) `python2.7 4.2.2.py capture.pcap`
- Your program MUST print out 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. For the purpose of this assignment, this rule applies even if the number of packets is very small. For example, following cases are all considered as attacks:

```
SYN=4 ACK+SYN=1
SYN=4 ACK+SYN=0
SYN=1 ACK+SYN=0
```

- Each IP address MUST be printed only once.
- Your program MUST silently ignore packets that are malformed or that are not using Ethernet, IP, and TCP.

Example output: A sample PCAP file captured from a real network can be downloaded at: `https://subversion.ews.illinois.edu/svn/sp16-ece422/_shared/mp4/lbl-internal.20041004-1305.port002.dump.anon.pcap`

(Source: `ftp://ftp.bro-ids.org/enterprise-traces/hdr-traces05`)

For this input, your program's output MUST 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
```

What to submit: Submit `4.2.2.py`, a Python program that accomplishes the task specified above. You SHOULD assume that the grader uses `dpkt` 1.8. You MAY use standard Python system libraries, but your program SHOULD otherwise be self-contained. We will grade your detector using a variety of different PCAP files.

Checkpoint 2: Submission Checklist

Inside your mp4 SVN repository, you will have auto-generated files named as below.
Make sure that your answers for all tasks up to this point are submitted in the following files by **Wednesday, December 14** at **6:00 PM**.

SVN Repository

Team Members

`partners.txt` : a text file containing NETIDs of both members, one NETID per line.
Place the NETID of the student making the submission in the first line.

example content of `partners.txt`

```
your_netid
partner_netid
```

Solution Format

example content of `4.2.1.1.txt`

```
cs46!
```

example content of `4.2.1.2.txt`

```
1.2.3.4
127.0.0.1
```

example content of `4.2.1.3.txt`

```
ftp
telnet
smtp
pop3
```

example content of `4.2.1.4.txt`

```
username1
p@ssw0rd123
```

example content of `4.2.1.5.txt`

```
1suPerSECretMesSageNKJn23kjsdjnK+Lskvnlksdf10dm23/sdfinvkwer2ThisShouldBe
InASingleLine23+4/
```

example content of `4.2.1.6.txt`

```
10
```

List of files that must be submitted for Checkpoint 2

- `partners.txt`
- `4.2.1.1.txt`
- `4.2.1.2.txt`
- `4.2.1.3.txt`
- `4.2.1.4.txt`
- `4.2.1.5.txt`
- `4.2.1.6.txt`
- `4.2.2.py`