# Project 3: Network Security

This project is split into two parts, with the first checkpoint due on **Wednesday, October 14** at **6:00pm** and the second checkpoint due on **Friday, October 23** at **6:00pm**. 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. Each semester everyone will be given ONE late extension that allows you to turn in up to one assignment up to 24 hours after the due date. Extensions are not automatic. So, if you want to use your late extension, you MUST send an e-mail to **ece422-staff@illinois.edu**. Late work will not be accepted after 24 hours past the due date.

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. Please find a partner as soon as possible. If have trouble forming a team, post to Piazza's partner search forum. Some hardware does not work with some of the tools needed for this project such as Aircrack (Checkpoint 2). 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 directory, following the submission checklist given at the end of each checkpoint. Details on the filename and submission guideline is listed at the end of the document.

---

*"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 will given under the folder called "mp3" in your SVN directory. We've also generated some empty files for you to submit your answers in. You MUST submit your answers in the provided files; we will only grade what's there!

## 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.

## 3.1 Checkpoint 1 (20 points)

### 3.1.1 Exploring Network Traces (15 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 packet trace from a sample network we set up for this assignment. You will search for specific vulnerable behaviors and extract relevant details using the Wireshark network analyzer (`http://www.wireshark.org`).

Get the network trace from `https://subversion.ews.illinois.edu/svn/fa15-cs461/_shared/mp3/3_1.pcap` and examine it using Wireshark. Provide concise answers to the following questions using submission format.

#### 3.1.1.1 MAC, IP address (5 points)

Multiple hosts sent packets on the local network.

1. What are their MAC addresses?
2. What are their IP addresses?

**What to submit:** Submit a text file named `3.1.1.1_mac.txt` that contains the MAC addresses of hosts, and a text file named `3.1.1.1_ip.txt` contains the hosts' IP addresses. Write one address per line in the same order for both MAC address and IP address. Refer to solution format and example under *Checkpoint 1 Submission Checklist*.

#### 3.1.1.2 FTP server (5 points)

One of the clients connects to an FTP server during the trace.

1. What is the DNS hostname of the server it connects to?
2. Is the connection using Active or Passive FTP?

**What to submit:** Submit a text file named `3.1.1.2_dns.txt` containing DNS hostname, and a text file named `3.1.1.2_ftp.txt` containing whether it is Active or Passive FTP.

#### 3.1.1.3 HTTPS connection (5 points)

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:

1. What is the domain name of the site the client is connecting to?

   **What to submit:** Submit a text file named `3.1.1.3_domain.txt` containing your answer.

2. During the TLS handshake, the client provides a list of supported cipher suites. List the cipher suites. Refer to the website with a list of known cipher suites table `http://www.thesprawl.org/research/tls-and-ssl-cipher-suites/`. Double check whether cipher suite name matches from the given page.

   **What to submit:** Submit a text file named `3.1.1.3_client.txt` where each line contains each cipher suite's name. Refer to the website with a list of known cipher suites table `http://www.thesprawl.org/research/tls-and-ssl-cipher-suites/`. Double check whether cipher suite name matches from the given page.

3. What cipher suite does the server choose for the connection?

   **What to submit:** Submit a text file named `3.1.1.3_server.txt` containing the corresponding cipher name.

#### 3.1.1.4 Facebook traffic analysis (5 points)

One of the clients makes a number of requests to Facebook.

1. Even though logins are processed over HTTPS, what is insecure about the way the browser is authenticated to Facebook?

   **What to submit:** Submit a text file named `3.1.1.4_insecurity.txt` containing your answer.

2. How would this let an attacker impersonate the user on Facebook?

   **What to submit:** Submit a text file named `3.1.1.4_impersonate.txt` containing your answer.

3. How can users protect themselves against this type of attack?

   **What to submit:** Submit a text file named `3.1.1.4_protect.txt` containing your answer.

4. What did the user do while on the Facebook site?

   **What to submit:** Submit a text file named `3.1.1.4_user.txt` containing your answer.

# Checkpoint 1: Submission Checklist

Inside your mp3 directory svn, you will have the auto-generated files named as below. Make sure that your answers for all tasks up to this point are submitted in the following files before **Wednesday, October 14** at **6:00pm**:

## SVN Directory

`https://subversion.ews.illinois.edu/svn/fa15-cs461/NETID/mp3`

## Team Members

`partners.txt` : a text file containing netIDs of both members, one netid per line. Place the student's netID, whose directory contain your project submission, at the top of the file.

**example content of partners.txt**

```
netid1
netid2
```

## Solution Format

**example content of 3.1.1.1_mac.txt**

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

**example content of 3.1.1.1_ip.txt**

```
1.2.3.4
127.0.0.1
```

**example content of 3.1.1.2_dns.txt**

```
dns1.illinois.edu
```

**example content of 3.1.1.3_domain.txt**

```
illinois.edu
```

**example content of 3.1.1.3_client.txt**

```
TLS_NULL_WITH_NULL_NULL
TLS_RSA_WITH_NULL_MD5
TLS_RSA_WITH_NULL_SHA
```

## List of solution files that must be submitted for checkpoint 1

- `partners.txt`
- `3.1.1.1_mac.txt`
- `3.1.1.1_ip.txt`
- `3.1.1.2_dns.txt`
- `3.1.1.2_ftp.txt`
- `3.1.1.3_domain.txt`
- `3.1.1.3_client.txt`
- `3.1.1.3_server.txt`
- `3.1.1.4_insecurity.txt`
- `3.1.1.4_impersonate.txt`
- `3.1.1.4_protect.txt`
- `3.1.1.4_user.txt`

## 3.2 Checkpoint 2 (100 points)

### 3.2.1 Network Attacks (65 points)

In this part of the project, you will experiment with network attacks by cracking the password for a WEP-encrypted WiFi network, decrypting an HTTPS connection, and recovering a simulated victim's username and password.

In the basement area of Siebel Center, near SC 0220, there is WiFi network named `cs461fa15` that is protected with WEP, a common but insecure crypto protocol. 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 a connection to a password-protected HTTPS server every few seconds. Your goal is to find the username and password of a client and log in to this website.

The tools and techniques you use are up to you, but each part has some suggestions to get you started. If you are having wireless issues and want to *rent USB wireless adapter, you can visit TA's office hours.* Rent period is 24 hours. However, if you are done with the wireless exercise early, you can post on Piazza and arrange a meeting with a TA to return it before the original return time. This will be greatly helpful to share the limited number of equipments with others.

**Attention online students:** you will be doing this exercise via remote machine with necessary tools pre-installed. Each student's username and password are given in SVN `mp3/remote_credentials.txt` file. Once connected, you will need `sudo` command to run given tools, especially `aircrack-ng`.

#### 3.2.1.1 WEP cracking (10 points)

First, you will need to crack the WEP encryption key for the network. There are many online tutorials and automated tools available to help you perform this task. We **strongly recommend installing Kali Linux** (available from `http://www.kali.org`) as USB bootable OS and using Aircrack-ng (`https://www.aircrack-ng.org/`). Aircrack-ng is pre-installed in Kali Linux, but if you decide to use other OS, separate installation is necessary.

The WEP cracking process usually involves generating network traffic to speed up the collection of data. For this project, we've made sure that there's sufficient traffic on the network for data collection. When following an Aircrack tutorial, you may not need the steps that involve generating traffic. Try to avoid those steps of generating traffic if possible.

**What to submit:** Submit a text file named `3.2.1.1_wep.txt` that contains the WEP key for the network.

#### 3.2.1.2 Network analysis (20 points)

Once you've cracked the WEP key, join the network and examine the traffic. We recommend using Kali Linux and Wireshark. Determine the IP addresses of the client and server, and carefully investigate any services running on these machines. Nmap (`http://nmap.org`) is a powerful tool for probing remote hosts.

**What to submit**

1. Submit a text file named `3.2.1.2_ip.txt` that contains the IP addresses of the server (first line) and a client (second line).

2. Submit a text file named `3.2.1.2_protocol.txt` that contains a list of the network protocols used by the services running on each machine; one protocol per line.

#### 3.2.1.3 Attacks (35 points)

In order to discover the client's username and password, you'll need to decrypt the HTTPS traffic. Wireshark can do this for TLS connections that don't use forward secrecy, if you can 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.

**What to submit**

1. Submit a text file named `3.2.1.3_credentials.txt` that contains the username (first line) and password (second line) for the HTTPS site the client loads;

2. Submit a text file named `3.2.1.3_secret.txt` that contains the secret contents of the website the client tries to load;

3. Submit a text file named `3.2.1.3_years.txt` that contains 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.

### 3.2.2 Anomaly Detection (35 points)

In 3.1.1, you manually explored a network trace. Now, you will programmatically analyze trace data to detect suspicious behavior. Specifically, you will be attempting to identity 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 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.

Your task is to develop a Python program that analyzes a PCAP file in order to detect possible `SYN` scans. You should use a library for packet manipulation and dissection: either `dpkt` or `scapy`. Both are available in most package repositories. You can find more information about `dpkt` at `https://code.google.com/p/dpkt/` and view documentation by running `pydoc dpkt`, `pydoc dpkt.ip`, etc.; there's also a helpful tutorial here: `http://jon.oberheide.org/blog/2008/10/15/dpkt-tutorial-2-parsing-a-pcap-file/`. To learn about `scapy`, visit `http://www.secdev.org/projects/scapy/`.

Your program will take one argument, the name of the PCAP file to be analyzed, e.g.:

```
python2.7 detector.py capture.pcap
```

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
```

**What to submit** Submit a Python program that accomplishes the task specified above, as a file named `3.2.2.py`. You should assume that `dpkt` 1.8 and `scapy` 2.3.1 are available, and 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 mp3 directory svn, you will have the auto-generated files named as below. Make sure that your answers for all tasks up to this point are submitted in the following files before **Friday, October 23** at **6:00pm**:

## SVN Directory

`https://subversion.ews.illinois.edu/svn/fa15-cs461/NETID/mp3`

## Team Members

`partners.txt` : a text file containing netIDs of both members, one netid per line. Place the student's netID, whose directory contain your project submission, at the top of the file.

**example content of partners.txt**

```
netid1
netid2
```

**example1 content of 3.2.1.1_wep.txt**

```
73616d706c65
```

or

**example2 content of 3.2.1.1_wep.txt**

```
73:61:6d:70:6c:65
```

**example content of 3.2.1.2_ip.txt**

```
1.2.3.4
127.0.0.1
```

**example content of 3.2.1.2_protocol.txt**

```
ftp
telnet
smtp
pop3
```

**example content of 3.2.1.3_credentials.txt**

```
username
p@ssw0rd123
```

**example content of 3.2.1.3_secret.txt**

```
supersecretmessage
```

**example content of 3.2.1.3_years.txt**

```
10
```

## List of solution files that must be submitted for checkpoint 2

- `partners.txt`
- `3.2.1.1_wep.txt`
- `3.2.1.2_ip.txt`
- `3.2.1.2_protocol.txt`
- `3.2.1.3_credentials.txt`
- `3.2.1.3_secret.txt`
- `3.2.1.3_years.txt`
- `3.2.2.py`