

Lab #3

Due date & Time 11:59pm on November 30th, 2007.

Late Policy: Late labs will not be accepted.

1 Overview

In this lab, you will have some experience of network security. The network packets are often logged for network forensics, protocol debugging and troubleshooting. Attackers often peer into the packets to get useful information. One can also do online network auditing to have real-time responses, or log the network traffic for off-line analysis.

In this lab, you will implement an off-line network traffic parser to find useful information. There are three parts of the project:

- Recover ftp username/password pairs from network traffic
- Get all the URLs a user visits from a browser
- Get all cookies contained in the network trace

Please read the handout carefully before starting this lab. You can use `c/c++`, and other utilities installed on the lab machines, like `grep`. If you plan to use any source code that is created by others, please contact TA. Turnin instructions will be sent to the mailing list.

You will work in teams of 2 people. Find your partner and send an email to TA with subject `cs426 team [lastname_lastname]`. In the body, it should include your names and emails.

2 Tasks

This section describes the tasks. The input/output details are in Section 3.

2.1 FTP Password

FTP communication is not encrypted. By looking at the network traffic, one can find out the username and password a user is using. The task is to recover all the username and password of ftp accounts from a network trace file.

To simplify the task, you can have the following assumptions:

- The FTP server is using port 21.
- There will be no protocol other than FTP running on port 21 in the trace file.

2.2 HTTP URLs

Since HTTP traffic is not encrypted, You can also look at the network packets and find out what URLs the user visits. The task is to find out all the URLs the browser has visited, by parsing the trace file. The assumptions are:

- The HTTP server is using port 80.
- There will be no protocol other than HTTP running on port 80 in the trace file.
- During the protocol, neither the server nor the client gets disconnected.

2.3 HTTP Cookies

HTTP Cookies may contain critical state information, such as authentication token. Since cookies are transmitted in clear-text, you can extract cookies from the packet traces. The task is to find out all the cookies contained in the trace file.

The assumption is the same as that in Section 2.2.

To make the task easier, you only need to extract the cookies sent out by the browser. Don't extract the cookies sent by the web server. Don't worry about the "domain" and "path" attributes for cookies (If you don't know what they are, just ignore them).

3 Implementation

All the functionalities will be implemented within a single executable file. The stub source code `parser.cc` and the make file is provided. The syntax of the tool is:

```
Usage: parser datafile w_dir func
```

Here *datafile* is the file name of the trace file. *w_dir* is the working directory, which would be used by *tcpflow*(see the stub code for details). Make sure the working directory is empty. *func* is the functionality, which can be `summary`, `ftpaccount`, `httpurl`, or `cookies`. `Summary` is a functionality already implemented in the stub source code, which can be used as an example.

For function `ftpaccount`, each line of the output is either

```
IP USER username
```

or

```
IP PASS password
```

IP is the FTP server IP address. The IP addresses should be sorted in alphabetical order. And within the same IP address, the output should be in the same order that the data appeared in the trace file. If a user logs in several times using the same username and password, then print duplicated username/password pairs(see the example output).

For function `httpurl`, when a client sends GET request to the server, the URL is printed out. Each line of the output is like:

```
src_ip -> dest_ip: url
```

Here `src_ip` is the client IP address, `dest_ip` is the server IP address, and `url` is the URL the client requested. Give the output by the alphabetical order of the client IP, then by the client port, then by the server IP, then by the server port, and finally by the order the URLs appearing in the trace file. You may need to combine the host information to the URL field in the GET request to get the absolute URL. (Note that the ports are not printed out in the output)

For function `cookies`, each of the output is:

```
src_ip -> dest_ip: host name value
```

Here `src_ip` is the client IP address, `dest_ip` is the server IP address, `host` is the host name the cookie corresponds to, `name` and `value` are the cookie content. Give the output by the alphabetical order of the client IP, then by the client port, then by the server IP, then by the server port, and finally by the order the cookies appearing in the trace file. Print all the duplicated tuples. You may need to combine the host and cookies fields in the HTTP request header to generate the output.

Hint: the format for the cookie filed in HTTP request header:

$$\textit{Cookie} : name_1 = value_1; name_2 = value_2; \dots; name_k = value_k$$

You dont have to implement a parser to parse all possible trace files. Most the trace files (6 out of 8) used in testing/grading will be provided, and some output files will also be provided. You just need to implement the basic functionality. You don't need to worry about boundary cases, such as keyword appearing in other places of the payload. Please compare your output to the sample output file carefully to make sure they are exactly the same. All the IP addresses and port number should be in the same format as the `tcpflow` filenames.

Tip: use the filter of `tcpflow` properly to get only the packets you are interested in.

Stub code can be downloaded from

<http://www.cs.purdue.edu/homes/zmao/lab3.tar>

See the readme file for information about example output. You can add files to the directory `src`, but don't change the directory structure.

Tip: backup code often.

4 Related Tools & Readings

tcpdump is a tool for capturing network packets. It uses libpcap library. The trace files used in this lab are collected using *tcpdump* and saved using the *-w* switch. Information can be found at

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

tcpflow is a tool for capturing the data in TCP connections. Information can be found at

http : //www.circlemud.org/jelson/software/tcpflow/

grep is useful to find a match in the file.

FTP is described in rfc959 at

http : //www.faqs.org/rfcs/rfc959.html

Section 4.1.1. ACCESS CONTROL COMMANDS is useful for the lab.

HTTP is described in rfc2616 at

http : //www.w3.org/Protocols/rfc2616/rfc2616.html

Section 9.3 and Section 14.23 is useful for the lab

HTTP cookies is described in rfc2965 at

http : //rfc.net/rfc2965.html

Section 3.3.4 is useful for the lab.

Tip: for most of the reading materials, you dont have to understand all the details. Have an overview of a tool/protocol first and use the manual as a reference later.