Homework 3

The Assignment

The goal of this assignment will be to create an HTTP(S) proxy server to monitor traffic coming from your mobile device. From this, you will be able to see the requests that different applications make and observe the data that is being sent over the network. In addition, you will be able to observe how different apps use and design their different APIs to properly perform their functions. As an extension, you will modify the responses for a specific application to simulate a malicious attack that can be performed through such means.

Simple HTTP Proxy Server

A proxy is simply a server which sits in between a client and a destination and acts as a middle-man between the two. The client sends the proxy server a request and the proxy server makes a request on behalf of the client to the destination. It then accepts the destinations response and passes it along to the client. By doing so, a proxy server is able to sniff all of the traffic that passes between the two parties.

Your proxy server will be functioning as a server, listening for incoming client requests at a specified port.

To implement a simple HTTP proxy server, you need to be able to perform the following operations:

1. Parse the HTTP request line from the client
    a. In an HTTP request message, there are request header and request content; and in the request header, the first line is called request line. Request line consists of request method, request URI and the protocol version.
    b. From the request URI you need to identify the server to contact.
    c. You are supposed to handle methods specified in protocol HTTP/1.1 in this assignment.
        i. Note: You don't need to handle CONNECT command in this part
2. Forward the request to the server
    a. Requests should be sent over to the server using socket
3. Return the HTTP response from the server to the client
    a. The response should be sent back to the client using the socket connection.

You should be able to run your proxy server by executing:

```
mproxy -p 8888
```

or

python mproxy.py -p 8888
Thus the server will be listening on port 8888.

After the proxy server is up and running, you should be able set it as the proxy server on your phone. If implemented correctly, the webpage rendered through proxy will look identical as if there were no proxy.
You can test your proxy by viewing urls such as:
http://www.example.com
http://cs.ucsb.edu

Your proxy should support the following commands.
-h or --help
Prints a synopsis of the application usage and exits with return code 0.
-v or --version
Prints the name of the application, the version number (in this case the version has to be 0.1), the author and exists, returning 0.
[-p port] or [--port port]
The port your server will be listening on. If the port you try to listen is already occupied, just try another.
[-n num_of_worker] or [--numworker num_of_worker]
For HTTP traffic, it often occurs that multiple requests are issued at the same time; therefore, you need to be able to handle concurrent requests using a thread pool.
This parameter specifies the number of workers in the thread pool used for handling concurrent HTTP requests. (default: 10)
[-t timeout] or [--timeout timeout]
The time (seconds) to wait before give up waiting for response from server. (default: -1, meaning infinite)
[-l log] or [--log log]
Logs all the HTTP requests and their corresponding responses under the directory specified by log.
Each request & response exchange should be stored in a separate file.
The name of the file should be [index]_[source_ip]_[servername].
For example, the 10th request you handled with your phone's ip being 123.234.123.234, requesting www.google.com, should be named "10_123.234.123.234_www.google.com"; placed under specified directory.
The content of the log file should be the query content and the response content separated by an empty line.

Man-in-The-Middle HTTPS Proxy Server

Your goal in this part is to extend your HTTP Proxy to be able to examine HTTPS traffic in plain text form.

HTTP traffic is exposed directly to proxies, while HTTPS connections are designed to be private even through a standard proxy, which only passes the encrypted data around without knowing the actual content. Because of the added privacy benefit of HTTPS it is increasingly used in mobile applications.

Its privacy can be compromised through a Man-in-The-Middle (MiTM) proxy.
An MiTM HTTPS proxy works by establishing two HTTPS connections. One is with the client, pretending to be the server; and one is with the server, pretending to be a client. To the client, instead of relaying the actual certificate, you will need to create one of your own.

A critical part of HTTPS protocol is trusts through certificates.
On your mobile device, stores a set of trusted root certificates owned by certificate authority (CA). During an HTTPS interaction, the server will present your phone with its certificate. Only certificates signed by one of these CAs are accepted.
Also, in the certificate, there are “Common Name”, and “Alternative Names” listed. A client will examine whether any of these names matches the domain it wishes to contact. When pretending to be a server, you need to fake a certificate with the same “Common Name” and “Alternative Names”, signed by a trusted CA.
Therefore, you need to generated a CA Certificate of your own, and install it onto your mobile device. Thus, all the fake certificates you generated and signed using your own CA certificate will be accepted by the mobile device.

To implement a MiTM HTTPS proxy server, you need to be able to perform the following operations:

1. Get server address. An HTTPS session will start with a CONNECT request intended for the proxy, providing address of the server which the clients wishes to connect to. You need to parse the request and respond to the client.
    a. Note: CONNECT request is purposefully designed to work with proxies. During HTTPS session without proxy, the client simply start with an SSL handshake, without providing information about the target server.
2. Get Server Name Indication (SNI). Sometimes it is not enough to know only the server address to contact. There might be multiple services hosted on the same server address. In Client Hello the field “Server Name Indication (SNI)” serves to identify exactly which service is the target.
   You need to understand the structure of a Client Hello message, and extract from it the SNI. Using the SNI, you can now pretend to be the client and perform SSL handshake with the server.
3. During the SSL handshake with server, a server certificate will be sent to the proxy, containing the ‘Common Name’ and ‘Alternative Names’ needed to generate a fake certificate.
4. With fake certificate generated, the proxy can then pretend to be the server and finish SSL handshake with the client.

5. At this point, both SSL handshakes are finished. The proxy needs to decrypt the client request, send the request message to the server (encrypted). And then decrypt the server's response, send the response message to the client (encrypted).

You can test your proxy by viewing web pages whose url starts with https or using mobile applications (e.g. periscope, yelp, whisper v4.5.3 or below).

After implementing your proxy, answer the following questions.
For periscope,

1. How to get a list of followers for a user
2. How to get a list of current broadcasts

For Yelp,

1. How to search for nearby Starbucks
2. How to bookmark a restaurant

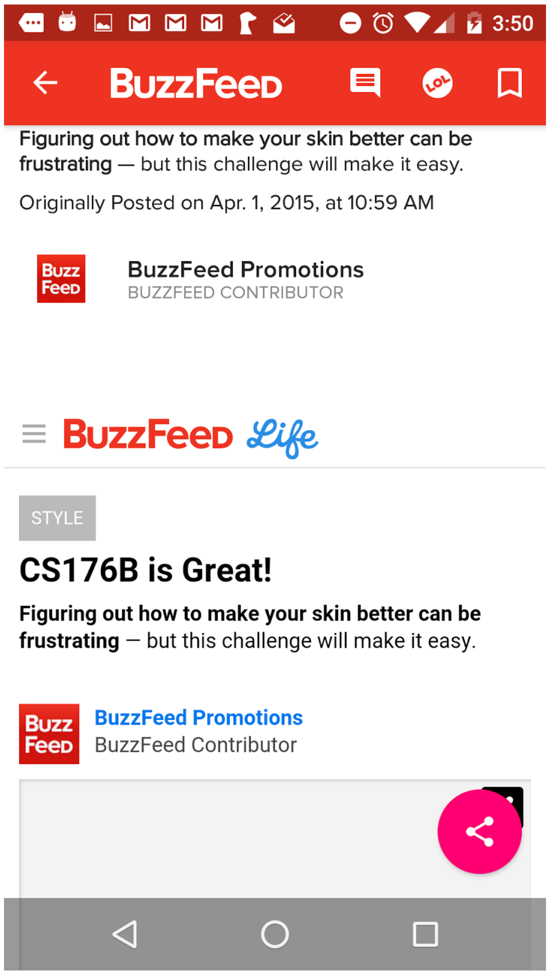
Write the answers to each question in README and include a log file of the matching exchange you captured.

Spoof/modify Traffic

This part aims at exploring what you can do with a proxy. To perform traffic spoofing, there are two steps involved. First, identifying the request / response we want to modify. Second, do it.

Here we presents four tasks you can accomplish using your proxy.

1. HTTP traffic modification
    a. For the application BuzzFeed, when you click into each article, the entire article is download through an HTTP request. Using your proxy, change the title of the article into "CS176B is Great!"
        i. Hint: a little knowledge of html is helpful
    b. For the application Whisper, each whisper is download as an image with an HTTP request. You should prepare an image containing the phrase "You are hacked"; and replace the image in the HTTP response with this one.
        i. Newer versions of Whisper uses non-HTTP(S) for some data transmission. Use versions v4.5.3 to make sure things work.


Figuring out how to make your skin better can be frustrating — but this challenge will make it easy.

Originally Posted on Apr. 1, 2015, at 10:59 AM

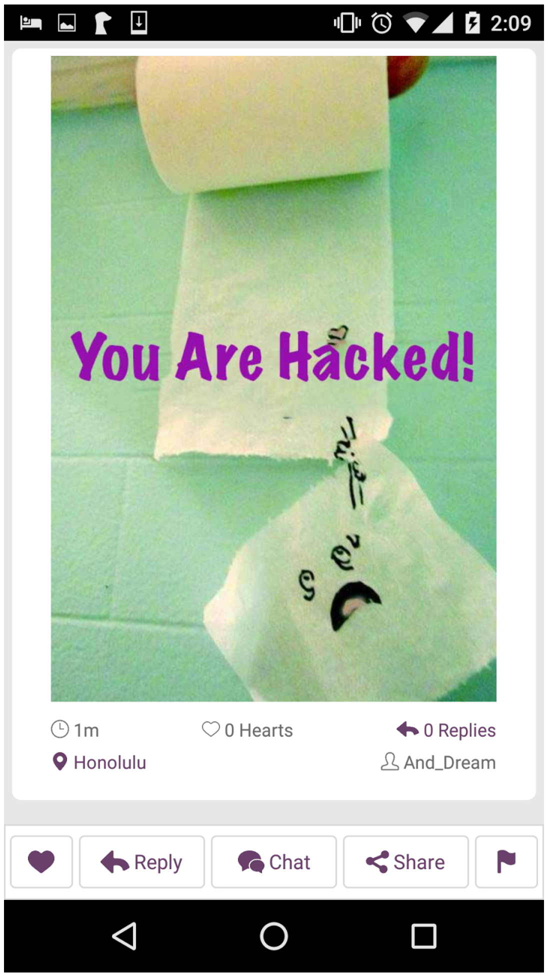
BuzzFeed Promotions
BUZZFEED CONTRIBUTOR

STYLE

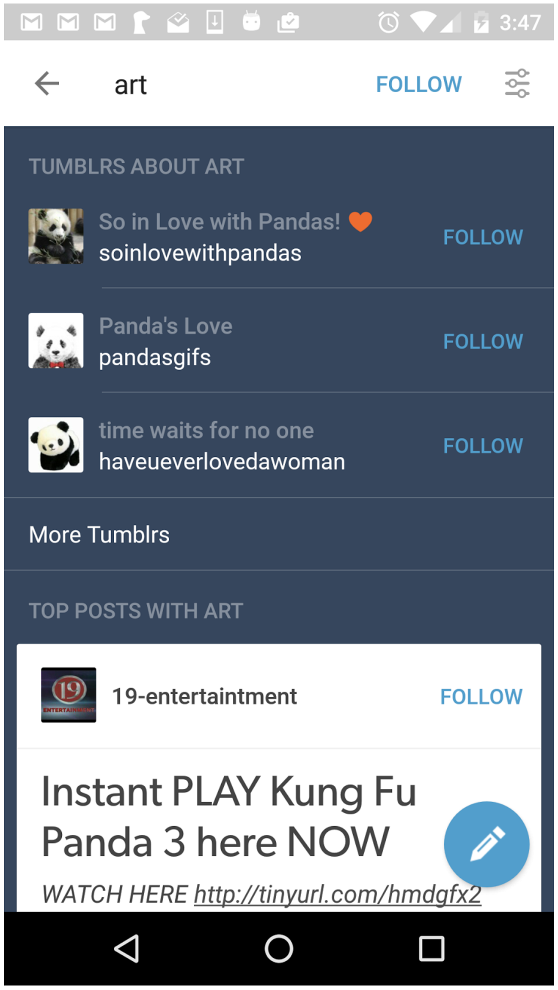
CS176B is Great!

Figuring out how to make your skin better can be frustrating — but this challenge will make it easy.

BuzzFeed Promotions
BuzzFeed Contributor



2. HTTPS traffic modification
    a. For the application Tumblr, when you use the search function, multiple HTTPS requests are issued. Change the search term in these HTTPS requests to “Panda”.
    b. For the application Periscope, change the number of viewer for each broadcasts to 1.



TUMBLRS ABOUT ART

So in Love with Pandas! ♥
soinlovewithpandas FOLLOW

Panda's Love
pandasgifs FOLLOW

time waits for no one
haveueverlovedawoman FOLLOW

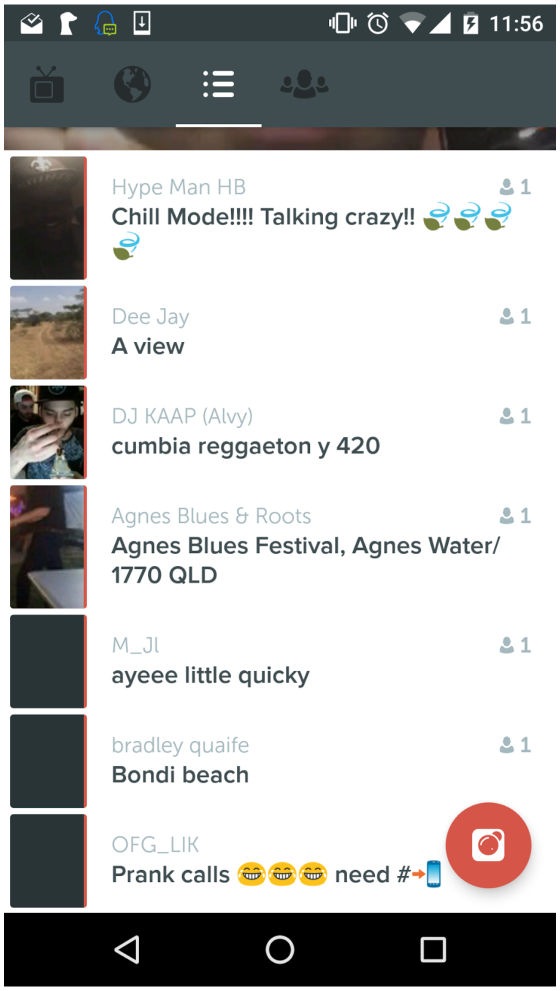
More Tumblrs

TOP POSTS WITH ART

19-entertaintment FOLLOW

Instant PLAY Kung Fu Panda 3 here NOW

WATCH HERE http://tinyurl.com/hmdgfx2



Hype Man HB 1
Chill Mode!!!! Talking crazy!! 🍃🍃🍃🍃

Dee Jay 1
A view

DJ KAAP (Alvy) 1
cumbia reggaeton y 420

Agnes Blues & Roots 1
Agnes Blues Festival, Agnes Water/ 1770 QLD

M_Jl 1
ayeee little quicky

bradley quaife 1
Bondi beach

OFG_LIK
Prank calls 😁😁😁 need #➡📱

You are welcomed to present other interesting use of the proxy :)

For each task, you need to provide in README how you implemented them. Including,

1. Reference to the function for identifying the requests you want to modify
2. Reference to the function for modifying the request / response.

Discussion

The application must be extremely resilient to wrong/unexpected input through the command line, as well as to unexpected traffic through the proxy.
You cannot assume all traffic received by the proxy will follow the standard protocol. In case of erroneous input, your proxy server cannot crash or exit. Instead you should return an error page whenever applicable.
Below is the list of common HTTP error codes you may issue.

1. 400 Bad Request: when the http request you get doesn't make sense
   Malformed request, incorrect server name, etc.

2. 502 Bad Gateway: when you as the proxy server received an invalid response from the backend server
3. 504 Gateway Timeout: when you as the proxy server did not receive a response from the backend server within the allowed time period.

Please understand that certain applications are not intended to be routed through proxy, e.g. banking apps; and certain apps have taken active measures against MiTM attack. So when you encounter error, don't panic. Just handle as many cases as you can.

Submission

The submission process for this assignment uses the turnin package. Do a man turnin to find more info about this program. The name of the assignment is hw3.
Your submission must contain one source file called mproxy.c or mproxy.py that servers as the entry point. In addition, you have to write a brief README file that describes your application (and possible issues and problems). You also have to include a file called AUTHORS that contains your name and your email address. Finally, if you are using c/c++ for this assignment, you need to include a Makefile file that describes how to build your application.

To submit:
Create a directory whose name is your CS account. For example, user John Doe --whose account is jdoe-- would do:

```
% mkdir jdoe
```

Put in the directory all files for homework 3.
Execute the turnin program. For example, user jdoe would execute:

```
% turnin hw3@cs176b jdoe
```

You can execute turnin up to 10 times per project. Earlier versions will be discarded. The timestamp of turnin has to be before the due date.