CSCD58H3 Winter 2018

Tutorial: 002

Week: 3

Date: January 23rd, 2018

Tools

1. **WireShark:**

   Network traffic analysis and protocol reverse engineering

2. **Telnet and nc:**

   Network swiss army knives. Can emulate various protocols and client and server models. Use their man pages to learn more.

3. **Traceroute:**

   Tracing packet path over a network at a given time. Use `man`pages to learn more.

Observing the HTTP protocol

- **The conversation as seen in Wireshark**

  0. Bring up wireshark and listen on the appropriate interface
  i. Navigate to `http://www.google.com` in your browser
  ii. After the page loads, stop the browsing
  iii. Filter for `HTTP` protocol communication by using `tcp.port == 80`
  iv. Follow TCP stream to view the entire communication

```
GET / HTTP/1.1
Host: www.google.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36 OPR/49.0.2725.64
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: SID=QAXIgdeg_mH_h2CmWs0naqGSKeQxnXH_QH1AIA4LCfrZuevTWmVE8qJ50O4adArBgNxLpQ.; HSID=AkgGOK1dsPvD3-1gH; APISID=zIquKQMQ2CQVe3zJ/A3TJnhp0k6FLk_DfE;
1P_JAR=2018-1-16-6; NID=121=sIOQZg4A6FszYKeSQWFOU92CMKHl0GWuZ4oLMRADgjJPjXGObVQnJN7MMzzI40p5ynC-
dRd6zVQmCFyK5iu5RFC6jn1fo01btf5TXtO-1l_xdBwkHHbxWOPOASmw7TIvxEPVydClsSHGlZqzs9JwoYAxxOJ2Ae_fJ2RmDP3GGchEqCZQ77VQwp_GJzKuf5GPPFyfC84CY8R-2yKV_-
vtzOSn3-83IU2d7hEM; SIDCC=AAiTGe_p8hSoVQREHYt0_yKnDvocDgkbT_WEq4E_SS0w9PZTGZt453M-tzM3zIDB7K8w0f-Y

HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Referrer-Policy: no-referrer
Location: http://www.google.ca/?gfe_rd=cr&dcr=0&ei=YLJmWtvFKdCfXtzLoegK
Content-Length: 266
Date: Tue, 23 Jan 2018 03:56:16 GMT

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.ca/?gfe_rd=cr&amp;dcr=0&amp;ei=YLJmWtvFKdCfXtzLoegK">here</A>.
</BODY></HTML>
```

- **Custom Webclient using `nc`**

We aim to achieve the same communication using `nc` as the web client

0. Open a connection to a HTTP connection to google's servers by running `nc www.google.com 80`

i. The command hangs for input

ii. enter `GET / HTTP/1.1` in the input and hit enter key twice. The reason is, HTTP protocol strings are terminated by double newline (`\r\n\r\n`).

iii. Compare the server's response witht he response as seen in wireshark.

```
KCs-MacBook-Pro:~ udonsi-kc$ nc www.google.com 80
GET / HTTP/1.1

HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Referrer-Policy: no-referrer
Location: http://www.google.ca/?gfe_rd=cr&dcr=0&ei=iLJmWryENtSfXoSgq5gN
Content-Length: 266
Date: Tue, 23 Jan 2018 03:56:56 GMT

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.ca/?gfe_rd=cr&amp;dcr=0&amp;ei=iLJmWryENtSfXoSgq5gN">
here</A>.
</BODY></HTML>
```

iv. Copy the request as seen in wireshark into a text file. Don't forget to add the double newline. Confirm you receive the same response as in wireshark.

```
KCs-MacBook-Pro:~ udonsi-kc$ nc www.google.com 80 < request.txt
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Referrer-Policy: no-referrer
Location: http://www.google.ca/?gfe_rd=cr&dcr=0&ei=BLVmWv-9NsyfXr_On-gJ
Content-Length: 266
Date: Tue, 23 Jan 2018 04:07:32 GMT

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.ca/?gfe_rd=cr&amp;dcr=0&amp;ei=BLVmWv-9NsyfXr_On-gJ">
here</A>.
</BODY></HTML>
```

Observing the SMTP protocol

- The conversation as seen in Wireshark

0. You'll require a native mail application. (Not a web browser)

i. Bring up wireshark and listen on the appropriate interface

ii. Attempt to send an email from a Gmail if you have an account.

iii. Filter for `smtp` protocol communication by using `smtp` as filter.

iv. Follow TCP stream to view the entire communication

```
220 smtp.gmail.com ESMTP 34sm4640488iom.88 - gsmtp
EHLO [138.51.80.138]
250-smtp.gmail.com at your service, [138.51.80.138]
250-SIZE 35882577
250-8BITMIME
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-CHUNKING
250 SMTPUTF8
STARTTLS
220 2.0.0 Ready to start TLS
...........Zf.....LY/......l..+......G..^....B...,.+.$.#.
.         ...0./.(.'...........k.g.9.3.......=.<.5./.
...........P.........smtp.gmail.com.
..................
.....................................S...O..Zf...p.l..;D.g.&|..F.p..XN.&.... .>B.v....
Z..........R....n/J.[../.....................w..K..u.`..Bi....f..~_.r....{.z......`.........H0F.!.............P.c...5...4..Q...l
.#.!...S.........c3.......'...aV..[;..w....+z
O. ....hp~.....\..=..........`.........H0F.!......tD.j..      ...eMKAo,......W3y!..!......P.][?6..
.o0...& ..";.....).....................0...0..j.......cbG.T...0
.         *.H..
.....0T1.0..U....US1.0...U.
..Google Trust Services1%0#..U....Google Internet Authority G30..
180110102839Z.
180404094100Z0h1.0   ..U....US1.0...U...
California1.0...U...
Mountain View1.0...U.
.
Google Inc1.0...U....smtp.gmail.com0.."0
```

- **Custom smtp client using `telnet`**

  We aim to achieve a similar communication using `telnet` as the SMTP web client.

  0. Launch the `telnet` command with no arguments (if installed). You'll be greeted with a `telnet` prompt.

  i. In order to send a mail, we need a mail server. We will be using the U of T mail server `t.mx.utoronto.ca`

  `telnet> o t.mx.utoronto.ca 25`

  ```
  udonsike@IITS-B473-01:~$ telnet
  telnet> o t.mx.utoronto.ca 25
  Trying 128.100.132.136...
  Connected to t.mx.utoronto.ca.
  Escape character is '^]'.
  220 bureau117 ESMTP Postfix
  ```

  compare the output in the terminal with the output in wireshark. The firstline in wireshark, beginning with `220 ...` confirms successful connection with the server.

  Note that the window hangs waiting for input

  ii. Say HELO to the server with the `EHLO` command

  `EHLO t.mx.utoronto.ca`

```
[EHLO t.mx.utoronto.ca
250-bureau117
250-PIPELINING
250-SIZE 52428800
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
```

The server responds (sometimes with a welcome message) and a list of available commands.

Compare the output in the terminal with the output in wireshark. See any similarities?

iii. The `STARTTLS` command shown in wireshark indicates the communication is now via encrypted channels (See wireshark). We will not be communicating using `STARTTLS`

iv. Sending an email ***(Perform while wireshark is listening to witness the communication packets)***:

To send an email we specify the sender and receipient:

a. `MAIL FROM: <sender email address here>` <- can be spoofed. For the sake of demonstration, the instructor has provided the address `fakeymcfakeface@mail.utoronto.ca`

b. `RCPT TO: <receiver email here>`

c. `DATA` <- This indicates the start of your message body

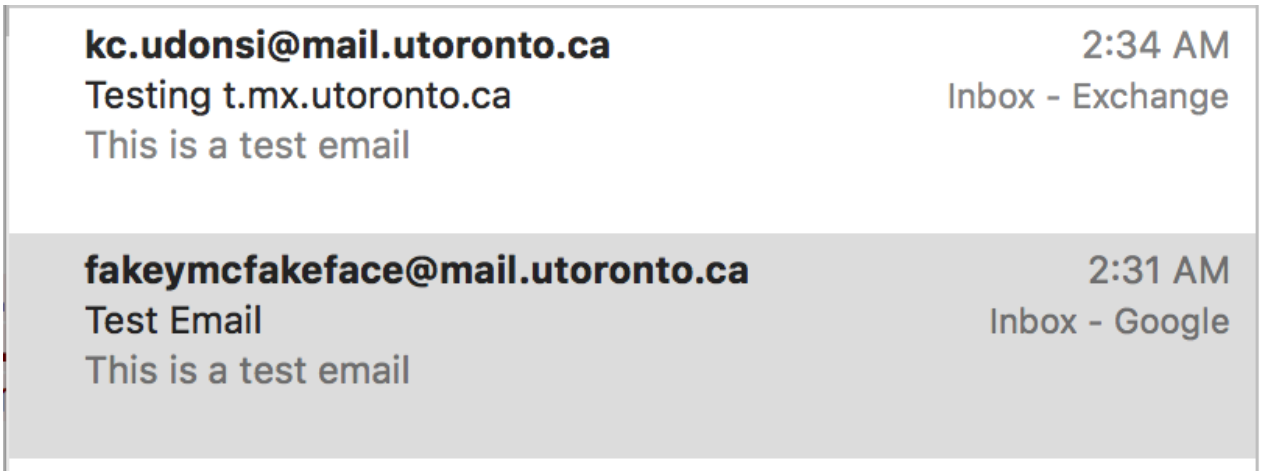
d. `SUBJECT:` (Optional) Specifying the email subject

e. `<email body goes here>`

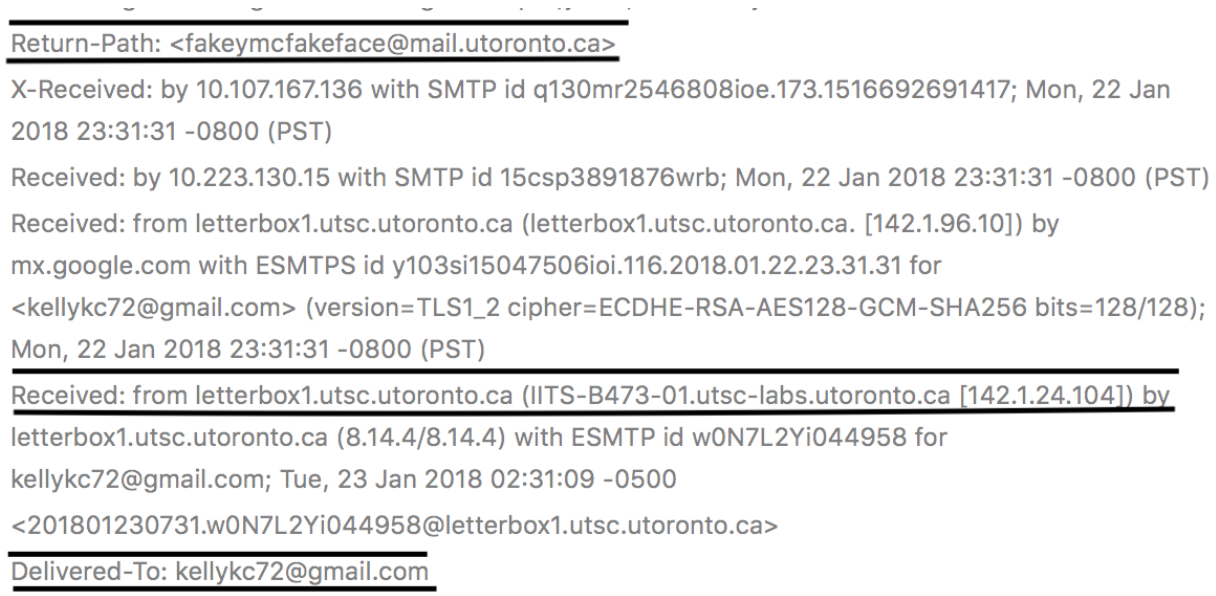
f. `.` <- A period indicating the end of the email

g. Hit enter

```
[MAIL FROM: kc.udonsi@mail.utoronto.ca
250 2.1.0 Ok
[RCPT TO: kc.udonsi@mail.utoronto.ca
250 2.1.5 Ok
[DATA
354 End data with <CR><LF>.<CR><LF>
[SUBJECT: Testing t.mx.utoronto.ca
[This is a test email
.
250 2.0.0 Ok: queued as 3zQg5F4D04z7t7N
```



h. Be advised. Except proper masking proxies are in place, the email is not really spoofed. The following image verifies the receiver is fully aware of the original sender's origin



Tracing the packet path

- Traceroute

  A network debugging tools that attempts to trace the path by which packets take to arrive at their destination. Nodes in the path are called `hops`. The tool discovers nodes in the path by incrementing `TTL` on each iteration such that the packets have just expired when they arrive a `hop`. The `hop`/`node` then sends back a packet to the tool saying the last received packet had expired. The tool counts these responses.

  Traceroute `www.google.com`

  i. Using the command `traceroute www.google.com`

  ii. Possible output

```
traceroute to www.google.com (172.217.1.164), 64 hops max
1   142.1.24.1  0.324ms  0.272ms  0.243ms
2   192.168.1.17  0.399ms  0.412ms  0.454ms
3   10.0.0.81  0.952ms  0.909ms  0.905ms
4   128.100.96.16  1.571ms  1.338ms  1.291ms
5   205.211.94.233  1.481ms  1.438ms  1.431ms
6   66.97.23.57  1.703ms  1.530ms  1.525ms
7   66.97.16.22  1.703ms  1.955ms  1.579ms
8   74.125.48.230  1.272ms  1.201ms  1.438ms
9   108.170.250.241  1.501ms  1.364ms  1.390ms
10   108.170.226.221  1.550ms  1.352ms  1.376ms
11   172.217.1.164  1.452ms  1.418ms  1.418ms
```

Each of the lines above indicate the IP that responded and the times taken to respond (`RTT`)since the tool sends three packets to each node. So it takes 11 hops from `iits-b473-01.utsc-labs.utoronto.ca` to `www.google.com`

ii. You can investigate the actual geolocation of those IPs by using:

```
#!/bin/bash

for i in $(traceroute $1 | grep -e ms | sed 's/\s\s*/ /g' | cut -d' ' -f3);
do
   echo;
   curl -s ipinfo.io/$i;
done
```

In the above script, we simply pass the ip addresses line by line to `curl ipinfo.io/`. A geoip tool.

iii. The output from the script can be sent to `less`

`sh script.sh www.google.com | less`