CS 370: Introduction to Security

05.09: Web security basics

Tu/Th 4:00 – 5:50 PM

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL

Secure AI Systems Lab

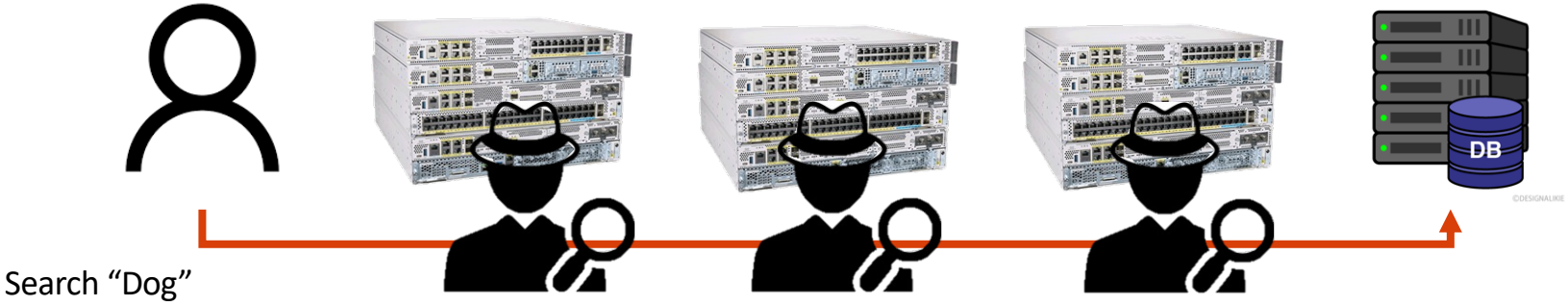
Some fun topics!

- Societal issues by large language models
  - https://mastodon.social/@danluu/110335983520055904
  - [Extra-credit opportunity: 5%] Write a CTF report with ChatGPT

Topics for today

- Recap: SSL and TLS security
  - SSL/TLS handshakes (hello-s)
  - (Perfect) Forward Security

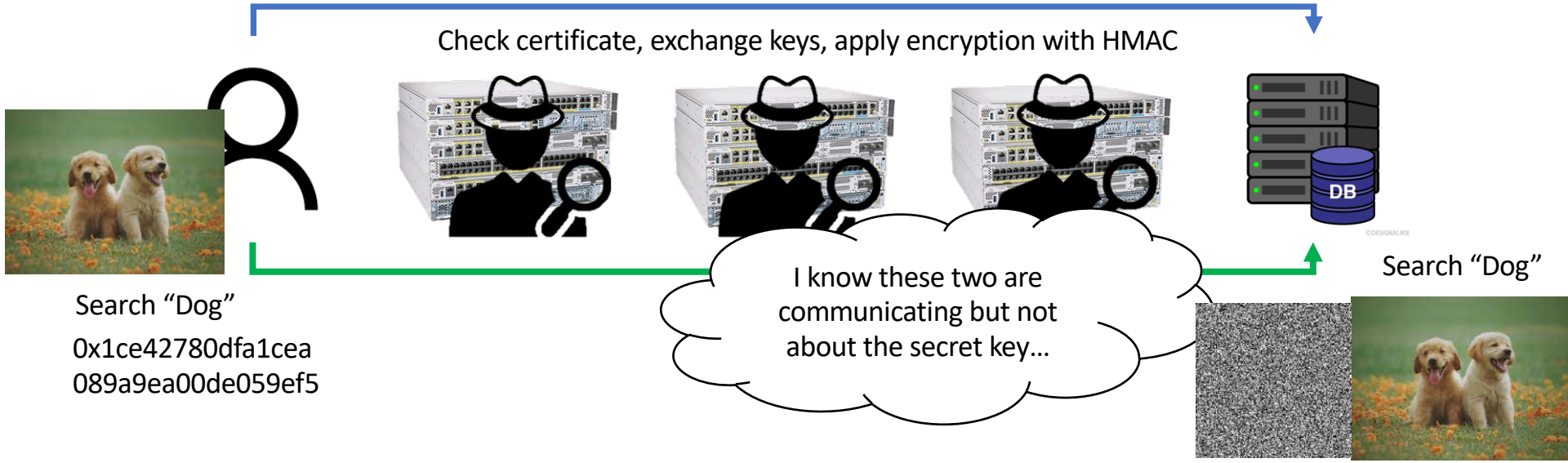
Recap: the Internet without security



Everybody in the Middle Knows That I Searched ‘dogs’ and They Also Know the Search Result… Ugh…

Recap: the Internet with a secure mechanism (SSL/TLS)

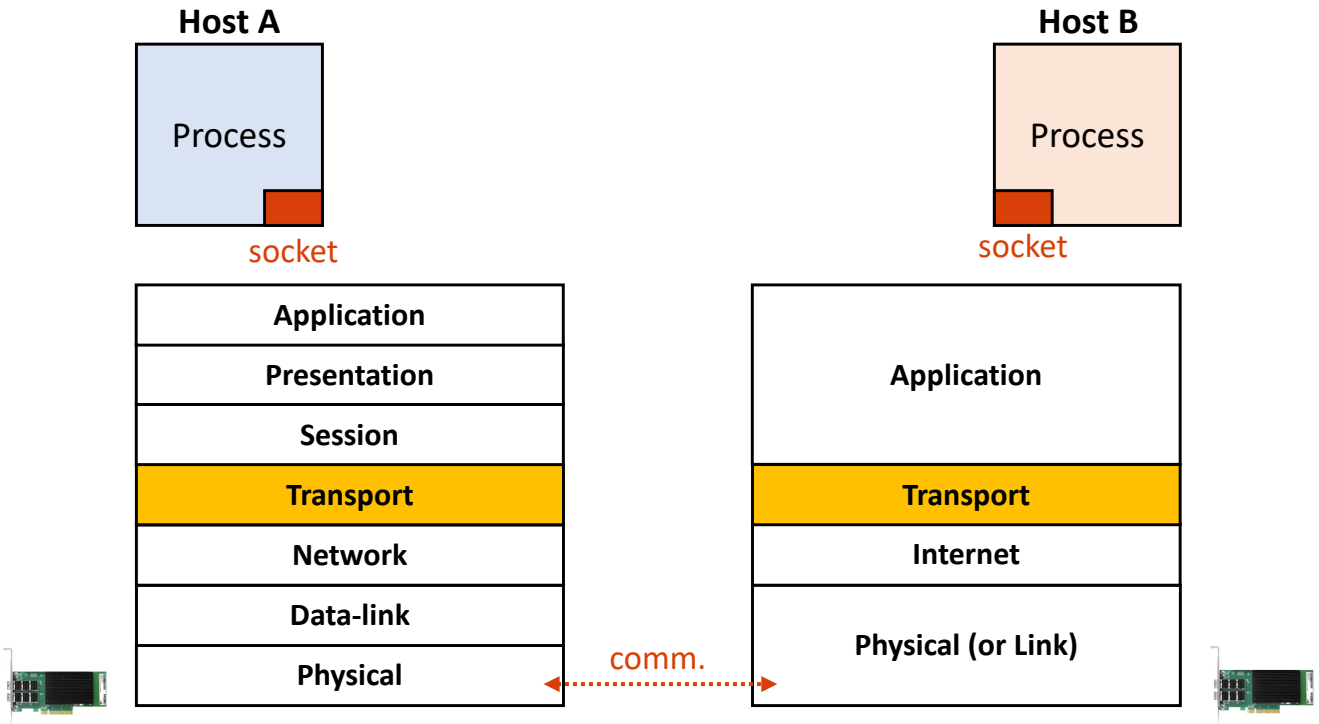
Middle mans never know
DH exchange keys!!



The Middlemen Will Only See the Encrypted Contents
They Will Never Know the Secret Key …

Recap: Why Transport Layer Security (TLS)?

- Independent from the application running on a host



Recap: SSL/TLS Handshaking

Client (You**)**

- 1. Send ‘client hello’
  - Version
  - Random number
  - Cipher suites available

Server (oregonstate.edu**)**

- 2. Send ‘server hello’
  - Version
  - Random number
  - Cipher suites chosen
- 3. Send ‘server certificate’
  - Full chain of digital certificates

Recap: SSL/TLS Handshaking

Client (You)

- 1. Send ‘client hello’

Server (oregonstate.edu)

- 2. Send ‘server hello’
- 3. Send ‘server certificate’
- 4. Server key exchange
  - Send ECDHE public values
  - Signed by the server’s private key
- 5. ‘server hello’ done

Recap: SSL/TLS Handshaking

Client (You**)**

- 1. Send ‘client hello’
- 6. Client key exchange
  - Send ECDHE public values (client)

Server (oregonstate.edu**)**

- 2. Send ‘server hello’
- 3. Send ‘server certificate’
- 4. Server key exchange
  - Send ECDHE public values
  - Signed by the server’s private key
- 5. ‘server hello’ done

Recap: SSL/TLS Handshaking

Client (You**)**

- 1. Send ‘client hello’
- 6. Client key exchange
- 7. Change cipher spec
- 8. Handshake message (encrypted)

Server (oregonstate.edu**)**

- 2. Send ‘server hello’
- 3. Send ‘server certificate’
- 4. Server key exchange
- 5. ‘server hello’ done
- 9. Change cipher spec
- 10. Handshake message (encrypted)

Now, We Can Start Communicating with Encrypted MSG!

Recap: SSL/TLS Handshaking

- Send/receive application data
  - Both client and server will send encrypted data
  - [encrypted data] [MAC]
    - Server: server_write_key and server_write_mac_key
    - Client : client_write_key and client_write_mac_key

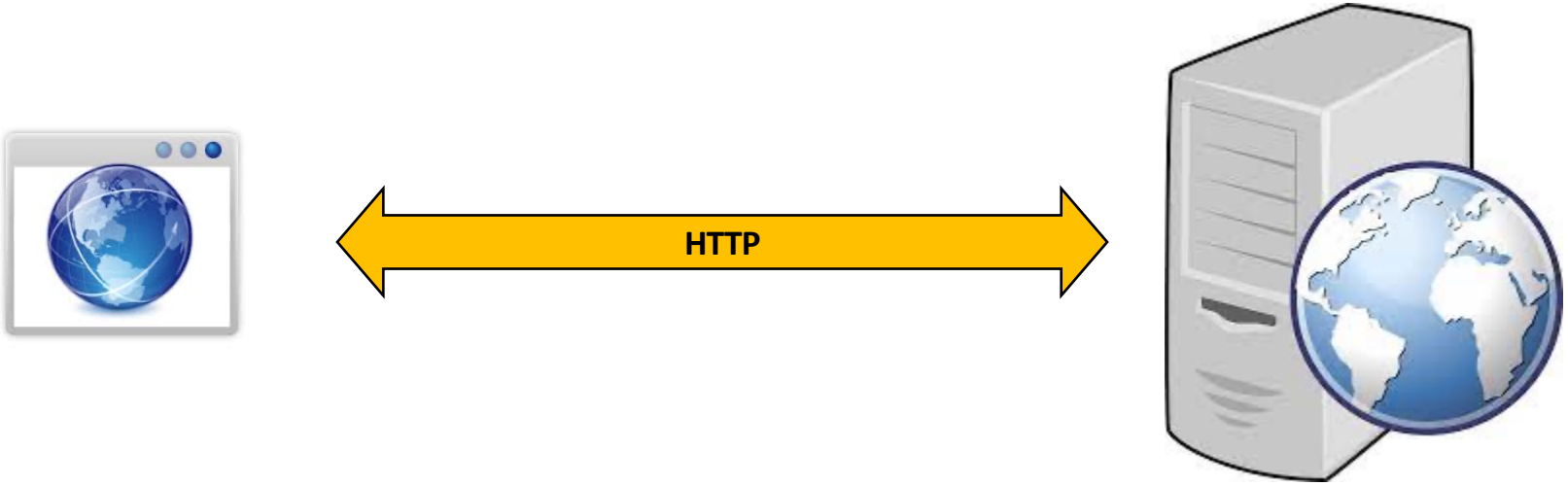
```
To generate the key material, compute

   key_block = PRF(SecurityParameters.master_secret,
                   "key expansion",
                   SecurityParameters.server_random +
                   SecurityParameters.client_random);

until enough output has been generated.  Then, the key_block is
partitioned as follows:

   client_write_MAC_key[SecurityParameters.mac_key_length]
   server_write_MAC_key[SecurityParameters.mac_key_length]
   client_write_key[SecurityParameters.enc_key_length]
   server_write_key[SecurityParameters.enc_key_length]
   client_write_IV[SecurityParameters.fixed_iv_length]
   server_write_IV[SecurityParameters.fixed_iv_length]
```

Topics for today

- Recap: SSL and TLS security
  - SSL/TLS handshakes (hello-s)
  - (Perfect) Forward Security
  - Example: a web-server with HTTPs

EXAMPLE: A WEB SERVER

- Suppose we talk to a webserver (HTTP)



Example: a web server

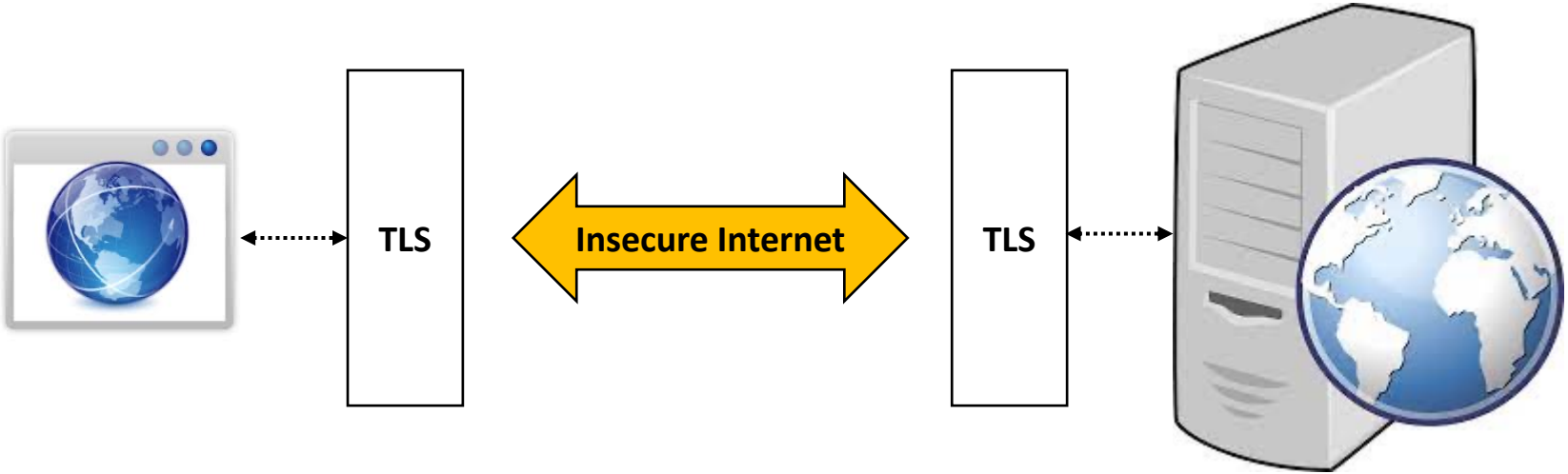
- Suppose we talk to a webserver (HTTP)

```
GET / HTTP/1.0
```


```
HTTP/1.0 200 OK
Date: Tue, 25 Oct 2022 12:53:12 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO
P3P: CP="This is not a P3P policy! S
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
```

EXAMPLE: A WEB SERVER

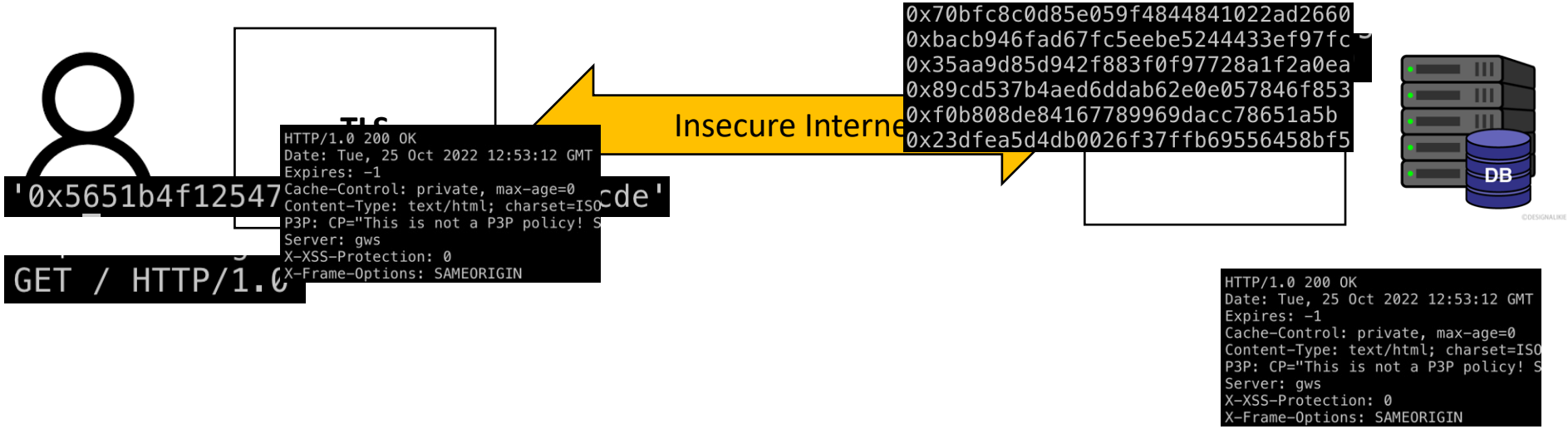
- Suppose we use HTTPs (instead of HTTP)



Example: a web server

Run **TLS handshake** to establish a secure channel

A Web Server Example



Let's see how HTTP packets look like

```
4  0.010756057    10.248.25.87          142.250.69.196        HTTP      144  GET / HTTP/1.1
```

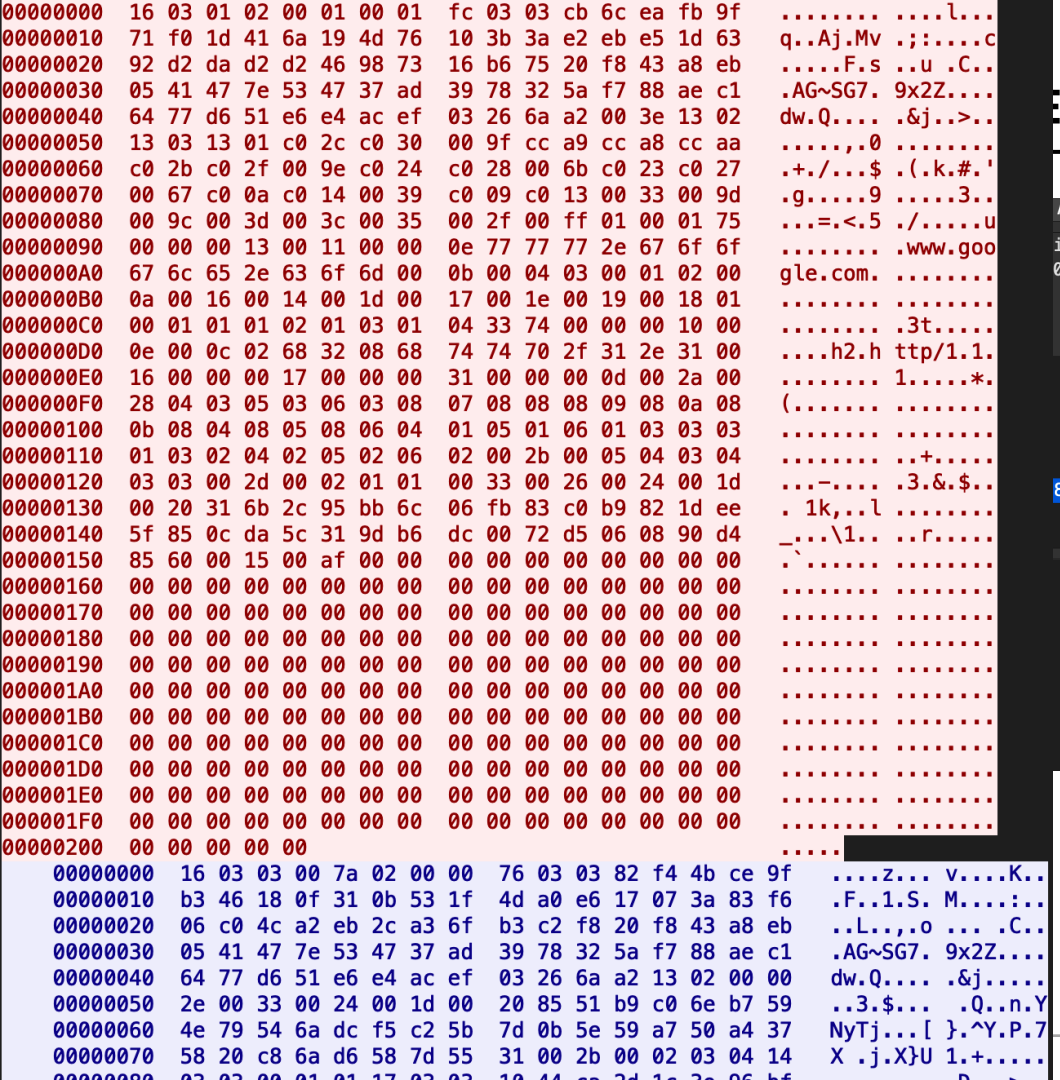
```
GET / HTTP/1.1
Host: www.google.com
User-Agent: curl/7.81.0
Accept: */*

HTTP/1.1 200 OK
Date: Tue, 25 Oct 2022 13:25:43 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2022-10-25-13; expires=Thu, 24-Nov-2022 13:25:43 GMT; path=/; domain=.google.com;
Secure
Set-Cookie: AEC=AakniGOAPvX70HdROvGjd5tdhzmMk-ZntDxb9jZGhAdPNSmqmwQc2AumlRI; expires=Sun, 23-Apr-2023
13:25:43 GMT; path=/; domain=.google.com; Secure; HttpOnly; SameSite=lax
Set-Cookie: NID=511=MkIeDpP817QKD-9oufZM9-
MAnHFlDpvagPc6jwK6l-2onKyCQID83aSymrg5ss1SUexUDpaSsNb9MrcxpnaXhezc9engEZrNmX4qgoG7Zodt4Fy-
HP9FQI6DbeY6GLGCma0MBOnUmze5m6Ys-i6jSvc6WFJUkye67iOSgFuG72c; expires=Wed, 26-Apr-2023 13:25:43 GMT;
path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked

348f
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta
content="Search the world's information, including webpages, images, videos and more. Google has many
special features to help you find exactly what you're looking for." name="description"><meta
content="noodp" name="robots"><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta
content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image"><title>Google</
title><script nonce="zhUe3NfmQtn_Ha4HtJSi3A">(function(){window.google={kEI:'1-
NXY8iGIs2T0PEP5fy9CA',kEXPI:'0,18167,1284369,56873,6059,206,4804,2316,383,246,5,5367,1123753,1197698,703
,302561,77529,16114,19398,9286,22431,1361,284,12036,17579,4998,13228,3847,6885,3737,22741,5081,1594,1278
,2742,149,1943,1983,214,4100,109,3405,606,2023,1777,520,14670,605,2622,2845,7,4808,791,28171,1851,2614,1
2710,432,3,1590,1,5444,149,11323,2652,4,1528,2304,7039,22023,5708,7356,16639,16808,1435,5827,2530,4094,1
7,4035,3,3541,1,14263,27894,2,14019,2373,342,4931,6470,9868,1755,5679,1021,2380,22668,6074,4568,6258,234
18,1252,5835,14968,4332,2204,5280,445,2,2,1,10956,15676,8155,7381,2,3,15965,873,6577,3048,10007,9,1921,5
784,3995,19130,12192,4832,17016,122,700,4,1,2,2,2,2,8652,107,821,4337,785,1765,978,3023,2756,3546,2,2017
,14,82,950,1758,168,1014,751,202,1866,125,6416,1,1015,51,2197,488,922,613,1323,346,109,364,466,683,899,2
```

Packet 4.1 client pkt, 9 server pkts, 1 turn. Click to select.

Entire conversation (16kB) | Show data as ASCII | Stream 0

Let's see how HTTPS packets look like

```
    40 4.482276498   10.248.25.87          142.250.69.196        TLSv1…    146 Change Cipher Spec, Application Data
> Frame 40: 146 bytes on wire (1168 bits), 146 bytes captured (1168 bits) on interface wlp0s20f3, id 0
> Ethernet II, Src: IntelCor_6c:c3:5c (98:2c:bc:6c:c3:5c), Dst: IETF-VRRP-VRID_01 (00:00:5e:00:01:01)
> Internet Protocol Version 4, Src: 10.248.25.87, Dst: 142.250.69.196
> Transmission Control Protocol, Src Port: 44148, Dst Port: 443, Seq: 518, Ack: 4303, Len: 80
v Transport Layer Security
  > TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  v TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
      Opaque Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 69
      Encrypted Application Data: 8684730f1612223931bb38393d0213d0b0e0dd0aa2ea6b908fab28f13c4ef8f8beba27d2…
      [Application Data Protocol: http-over-tls]

0000  00 00 5e 00 01 01 98 2c  bc 6c c3 5c 08 00 45 00   ··^····, ·l·\··E·
0010  00 84 73 f7 40 00 40 06  cd 6f 0a f8 19 57 8e fa   ··s·@·@· ·o···W··
0020  45 c4 ac 74 01 bb de 97  e7 2a d2 93 69 1c 80 18   E··t···· ·*··i···
0030  01 f5 f9 83 00 00 01 01  08 0a 09 e8 fa 1d 62 2d   ········ ·····b-
0040  cc 7d 14 03 03 00 01 01  17 03 03 00 45 86 84 73   ·}····· ····E··s
0050  0f 16 12 22 39 31 bb 38  39 3d 02 13 d0 b0 e0 dd   ···"91·8 9=·····
0060  0a a2 ea 6b 90 8f ab 28  f1 3c 4e f8 f8 be ba 27   ···k···( ·<N····'
0070  d2 67 e8 e4 2e 71 28 62  13 11 7d fb a1 58 fc 0c   ·g···q(b ··}··X··
0080  1d 5b da 7c 91 3f 6d 9f  bb 1d 6c 0b 67 ce 18 23   ·[·|·?m· ··l·g··#
0090  b9 d8                                             ··
```


00000000 16 03 01 02 00 01 00 01 fc 03 03 cb 6c ea fb 9f ....l...
00000010 71 f0 1d 41 6a 19 4d 76 10 3b 3a e2 eb e5 1d 63 q..Aj.Mv .;:....c
00000020 92 d2 da d2 d2 46 98 73 16 b6 75 20 f8 43 a8 eb .....F.s ..u .C..
00000030 05 41 47 7e 53 47 37 ad 39 78 32 5a f7 88 ae c1 .AG~SG7. 9x2Z....
00000040 64 77 d6 51 e6 e4 ac ef 03 26 6a a2 00 3e 13 02 dw.Q.... .&j..>..
00000050 13 03 13 01 c0 2c c0 30 00 9f cc a9 cc a8 cc aa .....,.0 ....
00000060 c0 2b c0 2f 00 9e c0 24 c0 28 00 6b c0 23 c0 27 .+./...$.(.k.#.'
00000070 00 67 c0 0a c0 14 00 39 c0 09 c0 13 00 33 00 9d .g.....9 .....3..
00000080 00 9c 00 3d 00 3c 00 35 00 2f 00 ff 01 00 01 75 ...=.<.5 ./.....u
00000090 00 00 00 13 00 11 00 00 0e 77 77 77 2e 67 6f 6f .... .www.goo
000000A0 67 6c 65 2e 63 6f 6d 00 0b 00 04 03 00 01 02 00 gle.com. ....
000000B0 0a 00 16 00 14 00 1d 00 17 00 1e 00 19 00 18 01 ....
000000C0 00 01 01 01 02 01 03 01 04 33 74 00 00 00 10 00 .... .3t.....
000000D0 0e 00 0c 02 68 32 08 68 74 74 70 2f 31 2e 31 00 ....h2.h ttp/1.1.
000000E0 16 00 00 00 17 00 00 00 31 00 00 00 0d 00 2a 00 .... 1.....*.
000000F0 28 04 03 05 03 06 03 08 07 08 08 08 09 08 0a 08 (....... ....
00000100 0b 08 04 08 05 08 06 04 01 05 01 06 01 03 03 03 ....
00000110 01 03 02 04 02 05 02 06 02 00 2b 00 05 04 03 04 ....+.....
00000120 03 03 00 2d 00 02 01 01 00 33 00 26 00 24 00 1d ...-.... .3.&.$..
00000130 00 20 31 6b 2c 95 bb 6c 06 fb 83 c0 b9 82 1d ee . 1k,..l ....
00000140 5f 85 0c da 5c 31 9d b6 dc 00 72 d5 06 08 90 d4 _...\1.. ..r.....
00000150 85 60 00 15 00 af 00 00 00 00 00 00 00 00 00 00 .`...... ....
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000001A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000001B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000001C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000001D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000001E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000001F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
00000200 00 00 00 00 00 00 .....
00000000 16 03 03 00 7a 02 00 00 76 03 03 82 f4 4b ce 9f ....z... v....K..
00000010 b3 46 18 0f 31 0b 53 1f 4d a0 e6 17 07 3a 83 f6 .F..1.S. M....:..
00000020 06 c0 4c a2 eb 2c a3 6f b3 c2 f8 20 f8 43 a8 eb ..L..,.o ... .C..
00000030 05 41 47 7e 53 47 37 ad 39 78 32 5a f7 88 ae c1 .AG~SG7. 9x2Z....
00000040 64 77 d6 51 e6 e4 ac ef 03 26 6a a2 13 02 00 00 dw.Q.... .&j.....
00000050 2e 00 33 00 24 00 1d 00 20 85 51 b9 c0 6e b7 59 ..3.$... .Q..n.Y
00000060 4e 79 54 6a dc f5 c2 5b 7d 0b 5e 59 a7 50 a4 37 NyTj...[}.^Y.P.7
00000070 58 20 c8 6a d6 58 7d 55 31 00 2b 00 02 03 04 14 X .j.X}U 1.+.....
Application Data
id 0
01)
8beba27d2…

```
00000000  16 03 01 02 00 01 00 01  fc 03 03 cb 6c ea fb 9f
00000010  71 f0 1d 41 6a 19 4d 76  10 3b 3a e2 eb e5 1d 63
00000020  92 d2 da d2 d2 46 98 73  16 b6 75 20 f8 43 a8 eb
00000030  05 41 47 7e 53 47 37 ad  39 78 32 5a f7 88 ae c1
00000040  64 77 d6 51 e6 e4 ac ef  03 26 6a a2 00 3e 13 02
00000050  13 03 13 01 c0 2c c0 30  00 9f cc a9 cc a8 cc aa
00000060  c0 2b c0 2f 00 9e c0 24  c0 28 00 6b c0 23 c0 27
00000070  00 67 c0 0a c0 14 00 39  c0 09 c0 13 00 33 00 9d
00000080  00 9c 00 3d 00 3c 00 35  00 2f 00 ff 01 00 01 75
00000090  00 00 00 13 00 11 00 00  0e 77 77 77 2e 67 6f 6f
000000A0  67 6c 65 2e 63 6f 6d 00  0b 00 04 03 00 01 02 00
000000B0  0a 00 16 00 14 00 1d 00  17 00 1e 00 19 00 18 01
000000C0  00 01 01 01 02 01 03 01  04 33 74 00 00 00 10 00
000000D0  0e 00 0c 02 68 32 08 68  74 74 70 2f 31 2e 31 00
000000E0  16 00 00 00 17 00 00 00  31 00 00 00 0d 00 2a 00
000000F0  28 04 03 05 03 06 03 08  07 08 08 08 09 08 0a 08
00000100  0b 08 04 08 05 08 06 04  01 05 01 06 01 03 03 03
00000110  01 03 02 04 02 05 02 06  02 00 2b 00 05 04 03 04
00000120  03 03 00 2d 00 02 01 01  00 33 00 26 00 24 00 1d
00000130  00 20 31 6b 2c 95 bb 6c  06 fb 83 c0 b9 82 1d ee
00000140  5f 85 0c da 5c 31 9d b6  dc 00 72 d5 06 08 90 d4
00000150  85 60 00 15 00 af 00 00  00 00 00 00 00 00 00 00
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00000200  00 00 00 00 00
    00000000  16 03 03 00 7a 02 00 00  76 03 03 82 f4 4b ce
    00000010  b3 46 18 0f 31 0b 53 1f  4d a0 e6 17 07 3a 83
    00000020  06 c0 4c a2 eb 2c a3 6f  b3 c2 f8 20 f8 43 a8
    00000030  05 41 47 7e 53 47 37 ad  39 78 32 5a f7 88 ae
    00000040  64 77 d6 51 e6 e4 ac ef  03 26 6a a2 13 02 00
    00000050  2e 00 33 00 24 00 1d 00  20 85 51 b9 c0 6e b7
    00000060  4e 79 54 6a dc f5 c2 5b  7d 0b 5e 59 a7 50 a4
    00000070  58 20 [illegible]
```

```
00000205  14 03 03 00 01 01 17 03  03 00 45 86 84 73 0f 16  ........ ..E..s..
00000215  12 22 39 31 bb 38 39 3d  02 13 d0 b0 e0 dd 0a a2  ."91.89= ........
00000225  ea 6b 90 8f ab 28 f1 3c  4e f8 f8 be ba 27 d2 67  .k...(.< N....'.g
00000235  e8 e4 2e 71 28 62 13 11  7d fb a1 58 fc 0c 1d 5b  ...q(b.. }..X...[
00000245  da 7c 91 3f 6d 9f bb 1d  6c 0b 67 ce 18 23 b9 d8  .|.?m... l.g..#..
00000255  17 03 03 00 29 ae d9 ce  dc e1 eb c5 15 ed ab 31  ....)... .......1
00000265  09 28 e9 65 87 98 4a 7a  76 e9 4b 19 f7 8a 12 d9  .(.e..Jz v.K.....
00000275  07 f3 87 8d 9d e1 dc 6e  af 3e 52 bd 94 81 17 03  .......n .>R.....
00000285  03 00 2c ff 1d 93 26 f3  b8 64 16 37 40 d9 4b 87  ..,...&. .d.7@.K.
00000295  56 6a 20 78 46 14 01 12  fd 1e f8 82 8e 01 44 53  Vj xF... ......DS
000002A5  b2 e6 c8 01 ca fe 25 86  d4 b4 39 1d 18 85 f9     ......%. ..9....
000002B4  17 03 03 00 1e 2d 05 11  4e c9 af f5 05 89 07 05  .....-.. N.......
000002C4  27 55 03 a0 0b 74 35 c7  25 d9 03 89 4e 97 87 70  'U...t5. %...N..p
000002D4  a0 ba 26 17 03 03 00 39  31 66 19 54 9b d1 e9 c5  ..&....9 1f.T....
000002E4  f4 bc 2f 43 ff 0d 91 be  e8 11 ef f9 90 35 07 7e  ../C.... .....5.~
000002F4  4c de 3e 05 b5 b6 2a 34  7b 83 9d b6 48 32 e5 a9  L.>...*4 {...H2..
00000304  17 12 f2 94 3c a2 27 2c  75 da 77 8f 98 71 6a 1d  ....<.', u.w..qj.
00000314  47                                                G
    000010CE  17 03 03 02 45 f8 f4 1d  68 b1 7e e5 a2 c6 1f ec  ....E... h.~.....
    000010DE  2a 27 d0 d9 cb 69 5d 4a  31 7b d4 54 43 e2 8f e7  *'...i]J 1{.TC...
    000010EE  e9 d0 d7 1e 8b 4f da 2a  8e 41 26 91 2a 27 d2 bc  .....O.* .A&.*'..
    000010FE  a9 de 8f 07 57 b5 72 01  11 2f 42 c4 e9 8f 41 80  ....W.r. ./B...A.
    0000110E  29 84 2b b7 8b db 8a a6  63 19 70 a3 c8 7c 28 85  ).+..... c.p..|(.
    0000111E  17 00 86 d0 ea 02 30 f3  1f 8e 6b a0 c9 19 77 de  ......0. ..k...w.
    0000112E  31 4f 61 e3 d8 4b 8e dc  c6 c7 f2 32 fa 70 f0 e1  1Oa..K.. ...2.p..
    0000113E  bb af 9c 79 e0 a9 f1 50  6c da d7 e2 36 eb 0b bb  ...y...P l...6...
    0000114E  09 f2 a3 7d a0 13 46 2e  3a 81 5c 77 d4 05 c5 2e  ...}..F. :.\w....
    0000115E  6f ba 65 49 52 1d f5 0b  1b 7d db c5 f9 1d ab ec  o.eIR... .}......
    0000116E  39 d3 40 0a 4b e3 f6 80  56 e2 e7 c5 d3 b8 df 79  9.@.K... V......y
    0000117E  b5 8f 07 48 61 30 a8 19  08 00 f5 51 d1 20 a6 b8  ...Ha0.. ...Q. ..
    0000118E  29 92 52 ae 46 89 ce 2d  43 a9 b1 ec 62 0f 69 f2  ).R.F..- C...b.i.
    0000119E  ff 34 67 5f 92 94 9f 9a  3d e6 36 0c 73 b9 8f 5a  .4g_.... =.6.s..Z
    000011AE  2c bb 91 24 fd 94 8f c4  72 f2 41 6a 49 86 f7 aa  ,..$.... r.AjI...
    000011BE  8e 17 16 c6 0e 48 92 cf  7b b3 a5 74 ee b6 f4 f4  .....H.. {..t....
    000011CE  cb 39 a6 f0 e1 15 a0 46  52 1c ab b9 ab ea d9 82  .9.....F R.......
    000011DE  fb a2 77 08 3d 05 65 20  18 7f e3 dd 44 f4 2b 38  ..w.=.e  ....D.+8
    000011EE  e7 23 9e 7f c6 29 83 dd  0b f0 e4 d0 b7 a9 fe 18  .#...).. ........
    000011FE  83 8f 77 cc 9f 88 42 df  ad a2 41 76 8f 16 38 4e  ..w...B. ..Av..8N
```

Benefit of using TLS

- TLS establishes a secure communication channel
  - Over the insecure Internet
  - Adds authenticity, confidentiality, and integrity to the channel
- Applications don't have to change their protocol
  - Just wrap the protocol with TLS
- How can we make HTTP secure?
  - Wrap with TLS: https://...

How can we use TLS?

- Many libraries are available
  - OpenSSL
  - libsodium
  - bouncycastle
  - SSL/TLS support in many other languages (Python, etc.)

- Just wrap with this
  - What does it mean???

Example: python webserver and client

- Suppose we do the followings:
  - Send ‘Hello’ to the server
  - Receive a number from the server
  - Add one to the number and send back to the server

Non-TLS: Client

- Suppose we ..
  - Send ‘Hello’ to the server
  - Receive a number from the server
  - Add one to the number
    and send this back to the server

```python
#!/usr/bin/env python3

import socket

def main():
    client_socket = socket.socket()
    client_socket.connect(('127.0.0.1', 31337))

    print("Sending Hello...")
    client_socket.send(b'Hello\n')

    b_number = client_socket.recv(5)
    s_number = str(b_number, 'utf-8')
    print("Received %s" % s_number.strip())

    i_number = int(s_number)
    i_answer = i_number + 1

    print("Returning %d" % i_answer)
    b_answer = bytes(str(i_answer), 'utf-8')
    client_socket.send(b_answer)

    result = client_socket.recv(10)
    print(result)

    client_socket.close()

if __name__ == '__main__':
    main()
```

Non-TLS: Server

- Suppose we ..
  - Send ‘Hello’ to the server
  - Receive a number from the server
  - Add one to the number
    and send this back to the server

```
#!/usr/bin/env python3

import os
import random
import socket
import sys

def main():
    server_socket = socket.socket()
    server_socket.bind(('127.0.0.1', 31337))
    server_socket.listen(10)

    while True:
        conn, addr = server_socket.accept()
        if os.fork():
            conn.close()
        else:
            message = conn.recv(6)
            if (message != b'Hello\n'):
                sys.exit(-1)
            print(b"Received %s" % message)
            number = random.randint(1000,9999)
            print("Sending %d" % number)
            conn.send(bytes(str(number)+'\n', 'utf-8'))
            message = conn.recv(5)
            print(b"Received %s" % message)
            if number+1 != int(str(message, 'utf-8')):
                conn.send(b"Incorrect\n")
                print("Incorrect")
            else:
                conn.send(b"  Correct\n")
                print("Correct")
            conn.close()
            sys.exit(0)


if __name__ == '__main__':
    main()
```

Non-TLS client vs TLS client

```python
#!/usr/bin/env python3

import socket

def main():
    client_socket = socket.socket()

    client_socket.connect(('127.0.0.1', 31337))

    print("Sending Hello...")
    client_socket.send(b'Hello\n')

    b_number = client_socket.recv(5)
    s_number = str(b_number, 'utf-8')
    print("Received %s" % s_number.strip())

    i_number = int(s_number)
    i_answer = i_number + 1

    print("Returning %d" % i_answer)
    b_answer = bytes(str(i_answer), 'utf-8')
    client_socket.send(b_answer)

    result = client_socket.recv(10)
    print(result)

    client_socket.close()

if __name__ == '__main__':
    main()
```

```python
#!/usr/bin/env python3

import socket
import ssl

def main():
    client_socket = socket.socket()

    context = ssl.create_default_context()
    context.check_hostname = False          # bad example
    context.verify_mode = ssl.CERT_NONE     # bad example

    client_socket.connect(('127.0.0.1', 31337))
    ssl_client_socket = context.wrap_socket(client_socket)

    print("Sending Hello...")
    ssl_client_socket.send(b'Hello\n')

    b_number = ssl_client_socket.recv(5)
    s_number = str(b_number, 'utf-8')
    print("Received %s" % s_number.strip())

    i_number = int(s_number)
    i_answer = i_number + 1

    print("Returning %d" % i_answer)
    b_answer = bytes(str(i_answer), 'utf-8')
    ssl_client_socket.send(b_answer)

    result = ssl_client_socket.recv(10)
    print(result)

    ssl_client_socket.close()

if __name__ == '__main__':
    main()
```

Non-TLS server vs TLS server

```python
#!/usr/bin/env python3

import os
import random
import socket
import sys

def main():
    server_socket = socket.socket()
    server_socket.bind(('127.0.0.1', 31337))
    server_socket.listen(10)

    while True:
        conn, addr = server_socket.accept()
        if os.fork():
            conn.close()
        else:
            message = conn.recv(6)
            if (message != b'Hello\n'):
                sys.exit(-1)
            print(b"Received %s" % message)
            number = random.randint(1000,9999)
            print("Sending %d" % number)
            conn.send(bytes(str(number)+'\n', 'utf-8'))
            message = conn.recv(5)
            print(b"Received %s" % message)
            if number+1 != int(str(message, 'utf-8')):
                conn.send(b"Incorrect\n")
                print("Incorrect")
            else:
                conn.send(b"  Correct\n")
                print("Correct")
            conn.close()
            sys.exit(0)


if __name__ == '__main__':
    main()
```

```python
#!/usr/bin/env python3

import os
import random
import socket
import ssl
import sys

def main():
    server_socket = socket.socket()
    server_socket.bind(('127.0.0.1', 31337))
    server_socket.listen(10)

    context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
    context.load_cert_chain('cert.pem', 'key.pem')
    ssl_server_socket = context.wrap_socket(server_socket, server_side=True)

    while True:
        conn, addr = ssl_server_socket.accept()
        if os.fork():
            conn.close()
        else:
            message = conn.recv(6)
            if (message != b'Hello\n'):
                sys.exit(-1)
            print(b"Received %s" % message)
            number = random.randint(1000,9999)
            print("Sending %d" % number)
            conn.send(bytes(str(number)+'\n', 'utf-8'))
            message = conn.recv(5)
            print(b"Received %s" % message)
            if number+1 != int(str(message, 'utf-8')):
                conn.send(b"Incorrect\n")
                print("Incorrect")
            else:
                conn.send(b"  Correct\n")
                print("Correct")
            conn.close()
            sys.exit(0)


if __name__ == '__main__':
    main()
```

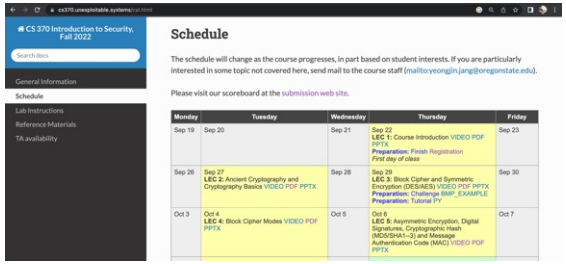
Topics for today

- Recap: SSL and TLS security
  - SSL/TLS handshakes (hello-s)
  - (Perfect) Forward Security
  - Example: a web-server with HTTPs
- Web security (authentication)
  - Password
  - Dictionary attack
  - SQL injection attack

WWW: WORLD-WIDE WEB

- WWW
  - **Formal:** An information system enabling documents and other web resources to be accessed over the Internet
  - **Informal:** the Internet for non-techie folks

- Uses HTTP as a document-delivery protocol
  - Request: GET /index.html HTTP/1.0\r\n
  - Response: 200 OK HTTP/1.0\r\n
  - … contents …



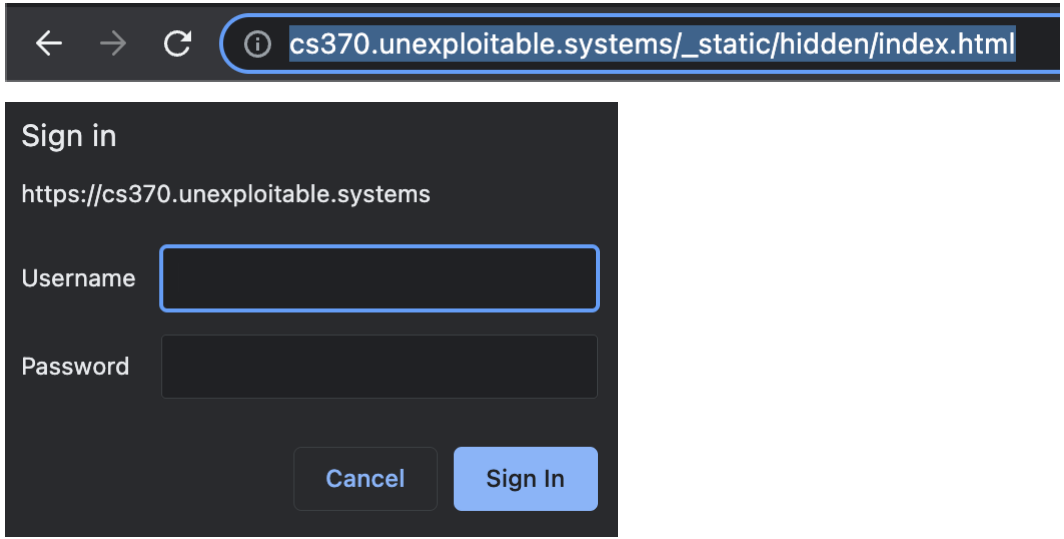
How can we do access control on the web?

- Suppose we don't have access control
  - Anyone can access any document via URLs (:= uniform resource locator)
  - http://www.bankofamerica.com/<your_account>

- We can apply access control on our websites
  - Use passwords
  - On the bankofamerica.com, type:
    - ID : your-account
    - PW: your-password

HTTP BASIC AUTHENTICATION

- HTTP basic authentication
  - A simple challenge and response mechanism
  - A server can request authentication information (ID and Password) from a client

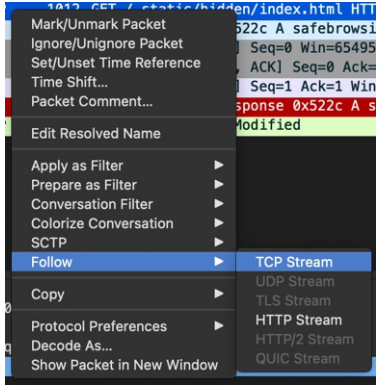


HTTP BASIC AUTHENTICATION: IN SECURE

- HTTP basic authentication
  - A simple challenge and response mechanism
  - A server can request authentication information (ID and Password) from a client

1	0.000000000	127.0.0.1	127.0.0.1	HTTP	1012	GET /_static/hidden/index.html HTTP/1.1
2	0.001963698	127.0.0.1	127.0.0.53	DNS	83	Standard query 0x522c A safebrowsing.google.com
3	0.002251748	127.0.0.1	127.0.0.1	TCP	74	53732 → 8080 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=1993954412 TSec…
4	0.002275826	127.0.0.1	127.0.0.1	TCP	74	8080 → 53732 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=1993…
5	0.002306468	127.0.0.1	127.0.0.1	TCP	66	53732 → 8080 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=1993954413 TSecr=1993954413
6	0.017663091	127.0.0.53	127.0.0.1	DNS	118	Standard query response 0x522c A safebrowsing.google.com CNAME sb.l.google.com A 142…
7	0.025120028	127.0.0.1	127.0.0.1	HTTP	254	HTTP/1.1 304 Not Modified

- Monitor the stream:




```
GET /_static/hidden/index.html HTTP/1.1
Host: cs370.unexploitable.systems:8080
Connection: keep-alive
Cache-Control: max-age=0
Authorization: Basic Ymx1ZTkwNTc6Y3MzNzB7QjRzSWNfQXVUaF9JNV9OMHRfczNDdVIzfQ==
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: _jsuid=1158429791;
experimentation_subject_id=eyJfcmFpbHMiOnsibWVzc2FnZSI6Iklqa3paak01WVdZekxUYzBOV0V0TkdJMU5pMDVZelpsTFRVd
05HUm1abVExTlRKak9DST0iLCJleHAiOm51bGwsInB1ciI6ImNvb2tpZS5leHBlcmltZW50YXRpb25fc3ViamVjdF9pZCJ9fQ%3D%3D-
-14df51e13094f383b80e4b21ff0c195dd82560ed; _jsuid=1158429791
If-None-Match: W/"6360b363-25"
If-Modified-Since: Tue, 01 Nov 2022 05:49:23 GMT

HTTP/1.1 304 Not Modified
Server: nginx/1.14.0 (Ubuntu)
Date: Tue, 01 Nov 2022 06:01:09 GMT
Last-Modified: Tue, 01 Nov 2022 05:49:23 GMT
Connection: keep-alive
ETag: "6360b363-25"
```

The string in the authorization field

- Uses Base64 Encoding
  - Binary to Text encoding
  - Uses printable 64-characters
- Suppose you have a string “ASD”
  - 01000001 01010011 01000100
  - 010000 010101 001101 000100 (6 bits)
  - Q V N E

```
>>> base64.b64encode(b"ASD")
b'QVNE'
```

Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/
Padding		=									

The string in the authorization field: binary to string

- Uses Base64 Encoding
  - Binary to Text encoding
  - Uses printable 64-characters
- Suppose you have a string “ffe0e8” (hex)
  - 11111111 11100000 11101000
  - 111111 111110 000011 101000 (6 bits)
  - / + D o

```
>>> base64.b64encode(b"\xff\xe0\xe8")
b'/+Do'
```

Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/
Padding		=									

Characteristics of a Base64 string

- Base64 encoding
  - All printable characters
  - Has / and + in addition to
  - [A-Za-z0-9]

- DIT (Micro-lab)
  - https://www.base64decode.net/
  - bmV1cm9u/b3ZlcmZsb3c6Y3MzNzB7Q+jRzSWNfQXV/USF9JNV9OMFRfczNDdVIzfQ==

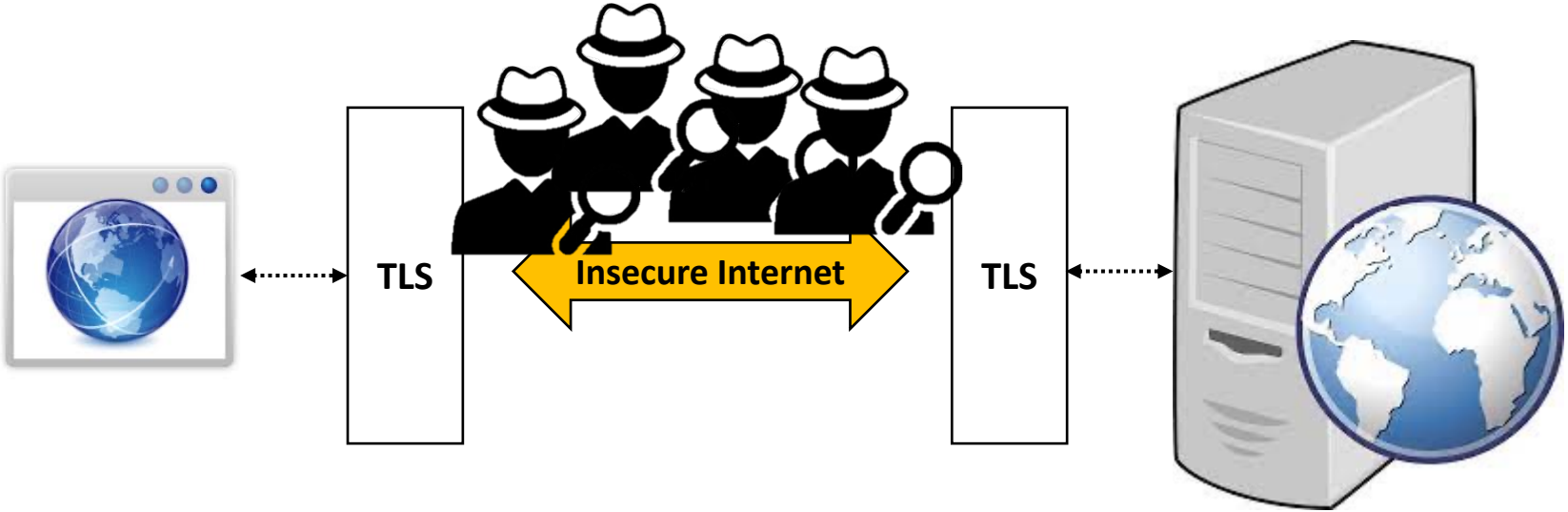
HTTP BASIC AUTHENTICATION: IMPLICATIONS

- We can use HTTP basic auth.
  - To do access control on our webpages
  - Users need to type the matching username and password
  - Otherwise, you can't access the page

- It is insecure:
  - HTTP packets are unencrypted
  - base64Encode(username:password) is there!

Solution: HTTPS

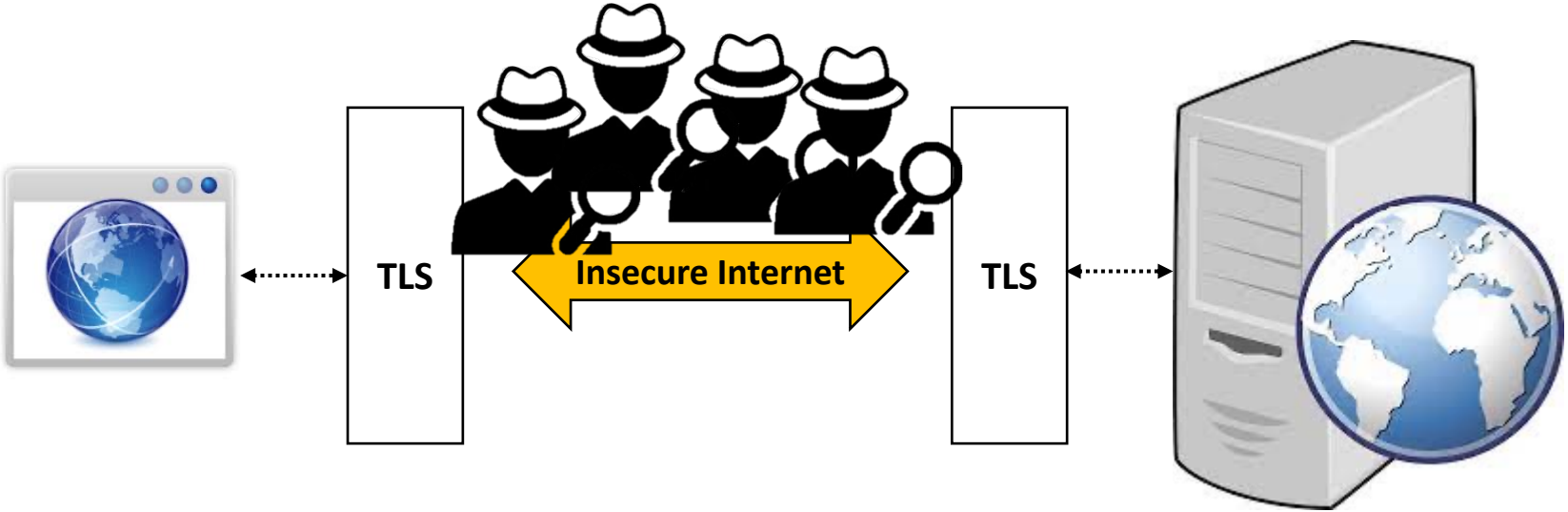
- Let's use HTTPS



- No one other than the server/client can see the content!

Solution: HTTPS

- Let's use HTTPS



- No one other than the server/client can see the content!

Are we safe now?

THE PASSWORDS WILL BE STORED TO THE SERVER

- So, anyone who can access the server can see them

Home › Email Security

Bed Bath & Beyond Invest
After Employee Falls for P

By Eduard Kovacs on November 01, 2022

Share Tweet Recommend 0 RSS

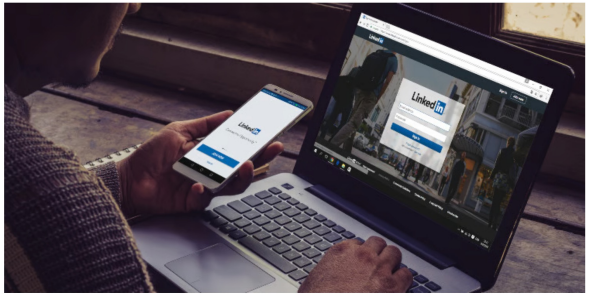
Bed Bath & Beyond revealed last week in an SEC filin
breach after an employee fell victim to a phishing at

This is not the first time Bed Bath & Beyond has disclosed
the retailer revealed that some customer accounts had b
hackers had obtained username and password combinatio
company and relied on the fact that many people use the
online accounts.

Scraped data of 500 million LinkedIn users being sold online, 2 million records leaked as proof

Updated on: 20 February 2023 9

Cybernews Team, Cybernews Team

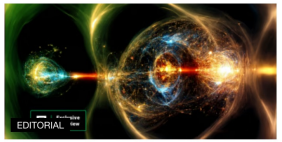


*Updated on 07/04: We updated our personal data leak checker database with **more than 780,000 email addresses associated with this leak**. Use it to find out if your LinkedIn profile has been scraped by the threat actors.*

Days after a massive Facebook data leak made the headlines, it seems like we're in for another one, this time involving LinkedIn.

An archive containing data purportedly scraped from 500 million LinkedIn profiles has been put for sale on a popular hacker forum, with another 2 million records leaked as a proof-of-concept sample by the post author.

Editor's choice



Quantum computing race explained: fast and furious

by Stefanie Schappert 05 May 2023

The World Economic Forum (WEF) published several think pieces this year describing a post-quantum computing world in which the global chasm between developed and underdeveloped populations only grows larger. But could the gloomy forecast be rosier than expected?

Read more

AI anxiety: the daunting prospect of mass unemployment

04 May 2023

Fake Instagram sugar daddy mimics Premier League mogul to

The passwords will be stored to the server

- So, anyone who can access the server can see them

Home › Email Security

Bed Bath & Beyond Invest
After Employee Falls for P

By Eduard Kovacs on November 01, 2022

Share Tweet Recommend 0 RSS

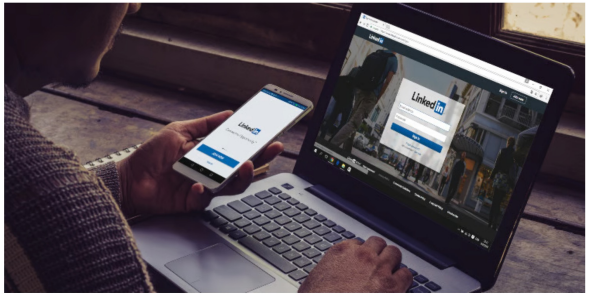
Bed Bath & Beyond revealed last week in an SEC filin
breach after an employee fell victim to a phishing at

This is not the first time Bed Bath & Beyond has disclosed
the retailer revealed that some customer accounts had b
hackers had obtained username and password combinatio
company and relied on the fact that
online accounts.

Scraped data of 500 million LinkedIn users being sold online, 2 million records leaked as proof

Updated on: 20 February 2023 9

Cybernews Team, Cybernews Team



Editor's choice

Quantum computing race explained: fast and furious

by Stefanie Schappert 05 May 2023

The World Economic Forum (WEF) published several think pieces this year describing a post-quantum computing world in which the global chasm between developed and underdeveloped populations only grows larger. But could the gloomy forecast be rosier than

Attackers put webservers on their radar; it can be **hacked**!
Passwords stored in the server **could also be leaked**

How can we address this issue?

- We can hide the passwords from the server
  - Do not store the passwords directly
  - Do store SHA256(“some_secret (salt)” + password)
  - Example:
    - SHA256(“some_secret (salt)” + “my-super-secure-password!@#$11”)
    - 59636881ab9bf34263cf3f4d90f25d2b91e74e8804b802d25c8f4bc5c80846ee

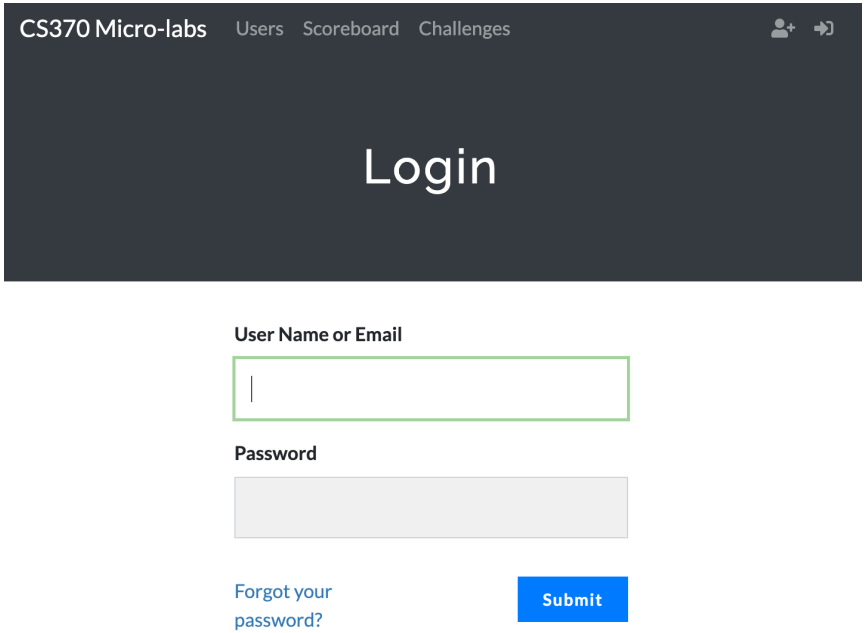
How can we address this issue? HASHING

- Hash the password
  - SHA256(“some_secret (salt)” + “my-super-secure-password!@#$11”)
  - 59636881ab9bf34263cf3f4d90f25d2b91e74e8804b802d25c8f4bc5c80846ee

- Can an adversary reconstruct the password from the hash?
  - SHA256
    - One-way function
    - Many x exists that satisfies H(x) = y
    - SHA256(‘Hello, world’) = SHA256(‘Something else’)
  - Good luck!

Secure practice: do NOT use HTTP basic auth

- Let's use the login form



SECURE PRACTICE: DO USE THE PASSWORD AUTHENTICATION

- Send ID/password but the server stores hash of the password

Let's think about how the server should work

- The server's procedure
  - A user enters their ID and password
  - The server queries the database to find (Username, SHA256('some secret' + password))
  - If exists, allow log-in
  - If not, then reject the request

Let's think about how the server should work

- The server searches the database
- Suppose the database uses SQL
  (Tutorials on: https://sqlbolt.com/lesson/select_queries_introduction)
  - SELECT (username, password) FROM users WHERE username = 'neuronoverflow' and password = SHA256(secret + "my-super-secure-password!@#$11")
  - Note:
    - The DB only stores the hash of the password
    - Suppose an adversary has access to the DB
    - They still need to compute the inverse to get the plaintext password

Security experiment: a brute-force attacker

- Suppose a powerful adversary
  - who breached the server
  - who has the entire database dump (:= all the password hashes)
  - who has the entire program source code (:= hashing algorithm and the salt)

- If it is a brute-force attacker:
  - Generate all the possible password combinations (plaintext)
  - Compare them with the hashes in the database
  - Example:
    - for string in all_the_candidates:
      - If SHA256('secret' + string) == '59636881ab9bf34263cf3f4d90f25d2b91e74e8804b802d25c8f4bc5c80846ee': print(string)

Security experiment: a brute-force attacker - cont'd

- Time it takes to run:
  - for string in all_the_candidates:
    - If SHA256('secret' + string) == '59636881ab9bf34263cf3f4d90f25d2b91e74e8804b802d25c8f4bc5c80846ee':
      print(string)
  - $\sim 2^{256}$ seconds (:= 1 second / a bit)

- Good luck!

Does it mean that we are secure?

- The security guarantee assumes
  - We choose the password randomly!

- In reality
  - (12345678) Easy to memorize and type
  - (OregonBeaverRocks) Some phrases familiar
  - (Oregon1234) Add numbers on the phrase
  - (password1234!!) Add special characters at the end
  - …

Does it mean that we are secure?

- The security guarante
  - We choose the pass
- In reality
  - (12345678) Easy to
  - (OregonBeaverRock
  - (Oregon1234) Add r
  - (password1234!!) A
  - …

Security experiment: dictionary attack

- Suppose that an adversary
  - Has a list of commonly used password
  - https://github.com/danielmiessler/SecLists/tree/master/Passwords/Common-Credentials

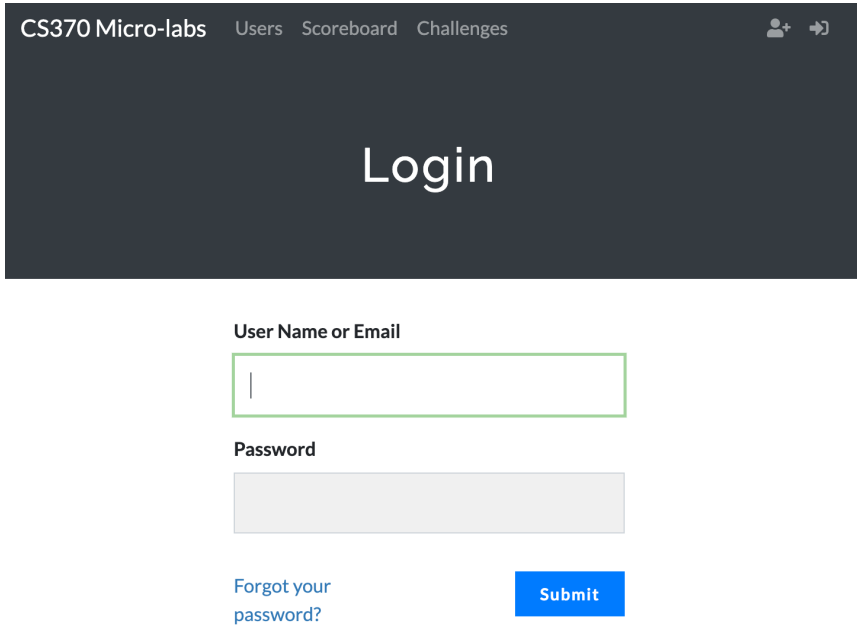
- Search space is significantly reduced
  - Suppose that the password is
    - 13 characters and consists of [A-Za-z0-9]
    - = 62^{13} possible combinations ($2.002854e^{23}$)
  - Suppose that
    - We know the password starts from 'Portland'
    - = 62^{5} possible combinations ($9.1613283e^{8}$)
    - = 10^{15} smaller

Security experiment: use special characters

- Suppose that the password is 8 characters
  - [A-Z] 26^8 = 208,827,064,576
  - [A-Za-z] 52^8 = 53,459,728,531,456
  - [A-Za-z0-9] and special chars = 95^8 = 6,634,204,312,890,625
    - 6,634 trillion cases
    - Good luck!

Brute-force is difficult; can we exploit the system?

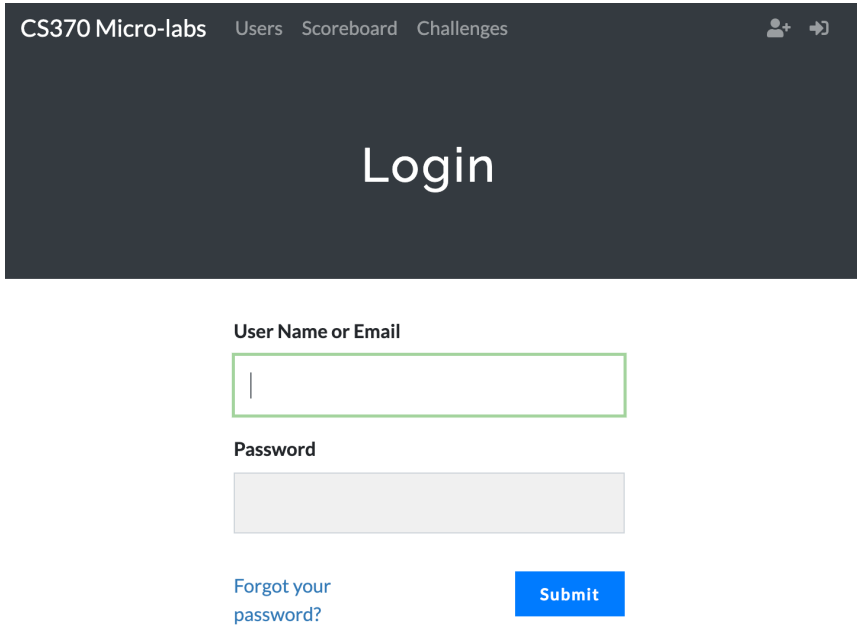
- Suppose the database uses SQL
  - SELECT (username, password) FROM users WHERE username = ‘neuronoverflow’ and password = SHA256(secret + “my-super-secure-password!@#$11”)



Can we exploit the system? SQL injection

- Suppose the database uses SQL
  - SELECT (username, password) FROM users WHERE username = ‘neuronoverflow’ and password = SHA256(secret + “my-super-secure-password!@#$11”)

- What if
  - We supply ‘or ‘a’=‘a as a password?
  - SELECT (username, password) FROM users WHERE username = ‘neuronoverflow’ and password = ‘’ or ‘a’ =‘a’
  - THIS IS ALWAYS TRUE!!!



Can we exploit the system? SQL injection

- What if we supply ‘or ‘a’=‘a as a password?
  - SELECT (username, password) FROM users WHERE username = ‘neuronoverflow’ and password = ‘’ or ‘a’ =‘a’

- This allows us:
  - To bypass password checking logic
  - By injecting carefully-crafted malicious data to the database SQL query

Can we exploit the system? SQL injection

- What if we supply ‘ union select (‘admin’, ‘a’) where ‘a’=‘a as a password?
  - SELECT (username, password) FROM users WHERE
  - username = ‘neuronoverflow’ and password = ‘’ union select (‘admin’, ‘a’) where ‘a’=‘a’

- How does it work?
  - None for the first select statement
  - and the 2nd statement will query
    - Username = ‘admin’
    - Password = ‘a’
    - Always return true ‘a’ = ‘a’

Topics for today

- Recap: SSL and TLS security
  - SSL/TLS handshakes (hello-s)
  - (Perfect) Forward Security
  - Example: a web-server with HTTPs
- Web security (authentication)
  - Password
  - Dictionary attack
  - SQL injection attack

Thank You!

Tu/Th 4:00 – 5:50 PM

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL

Secure AI Systems Lab