SlideSet #19: HTTP and HTTPS

http://www.garshol.priv.no/download/text/http-tut.html

Client / Server Big Picture

Client

Request →

Server

← Response

Language for these communications?
HTTP Protocol Stack

<table>
<thead>
<tr>
<th>HTTP</th>
<th>FTP</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TCP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Various</td>
<td></td>
</tr>
</tbody>
</table>

What does HTTP request look like?

- User enters URL:
  http://www.usna.edu/
- Browser sends request to www.usna.edu:
  GET / HTTP/1.0
  User-Agent: Mozilla/3.0 (compatible; Opera/3.0; Windows 95/NT4)
  Accept: */*
- What would GET line be if URL were...
  http://www.usna.edu/cs/news.html
What does HTTP response look like?

- If okay, server sends back response:
  
  ```text
  HTTP/1.1 200 OK
  content-length: 4303
  accept-ranges: bytes
  server: Apache/2.0.54 (Unix) PHP/5.0.4
  last-modified: Tue, 13 Aug 2013 13:18:07 GMT
  connection: close
  etag: "328f-10cf-1c8181c0"
  x-pad: avoid browser bug
  date: Wed, 14 Aug 2013 17:58:32 GMT
  content-type: text/html
  
  <!DOCTYPE html>
  <html> <head> ... 
  ```

Variants of the HTTP request

- **HEAD / HTTP/1.0**
- **GET /cgi-bin/query.pl?str=dogs&lang=en HTTP/1.0**
- **POST /cgi-bin/query.pl HTTP/1.0**
  
  ```text
  Content-Type: application/x-www-form-urlencoded
  Content-Length: 16
  str=dogs&lang=en
  ```

- **GET /img1.jpg HTTP/1.1**
- **GET /img6.jpg HTTP/1.1**
  
  ```text
  Host: www.host1.com
  Connection: close
  ```

Variants of the HTTP response

- Status codes
  - 200 OK
  - 301 Moved permanently
  - 400 Bad request
  - 403 Forbidden
  - 404 Not found
  - 500 Internal server error
  - 503 Service unavailable

Exercise: How do the HTTP request and response look like?

http://mope.academy.usna.edu/~adina/welcome.py?username=ac

```python
welcome.py
#!/usr/bin/env python3
from http import cookies
import urllib.parse
import cgi

# get parameters
params = cgi.FieldStorage()
username = params.getvalue("username")

cookie = cookies.SimpleCookie()
cookie["Username"] = urllib.parse.quote(username)

print (cookie)
print( "Content-type:text/html\n")
print ("<html><head><meta charset = "utf-8">
<title>Storing cookies with Python</title>
</head><body>"
print (<"h1">Welcome " + username + "</h1">);
print("<body>"<html3>");
```
Lab Exercise

• How does the HTTP request look like for

• Use netcat (nc) to request the page above
  – nc www.usna.edu 80
  – [The HTTP request here + 1 empty row]
  – What is the answer?

• Use openssl to request the page above using https
  – openssl s_client -connect www.usna.edu:443
  – [The HTTP request here + 1 empty row]
  – What is the answer?

Lab Exercise:
1. In public_html, create welcome.py with the code below
2. How do the HTTP request and response look like? Use nc to verify!

```python
# /var/lib/env python3
from http import cookies
import urllib.parse
import cgi

# get parameters
params = cgi.FieldStorage()
username = params.getvalue("username")

cookie = cookies.SimpleCookie()
cookie["Username"] = urllib.parse.quote(username)

print (cookie)
print ("Content-type:text/html\n"
)
print ("<!DOCTYPE html>
<html><head><meta charset = "utf-8">
<title>Storing cookies with Python</title>
<body>
<h1>Welcome " + username + "</h1>
</body></html>"
)
```

http://mope.academy.usna.edu/~mXXXXXX/welcome.py?username=ac
Lab Exercise

• Open Chrome → Tools → Developer Tools → Network tab

• Type http://mope.academy.usna.edu/~mXXXXXX/welcome.py?username=ac in the address bar (XXXXXX – your alpha)
  – What http request is made?
  – Check the content of the request by right-clicking on the path in the list and choose “Copy the request headers” – paste it into Notepad++

• Type www.google.com in the address bar
  – What http requests are made? Why?

• Type www.amazon.com in the address bar
  – How many requests are made?

HTTPS: HTTP over TLS

<table>
<thead>
<tr>
<th>HTTP</th>
<th>FTP</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS (SSL)</td>
<td>TCP</td>
<td>IP</td>
</tr>
<tr>
<td>Various</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Encryption (Covered in SY308)

- Encode / decode the data
- Need:
  - algorithm
  - key

- Symmetric encryption

- Asymmetric encryption
  - Interesting property: you can switch encryption and decryption key and get the same results

TLS Protocol Highlights (covered in SY308)

- Browser connects to SSL-enabled server
- Computers agree on encryption method
- Server sends its digital certificate (contains the public key)
- Browser and server generate session key
- Further communications are encrypted using the session key
HTTP over TLS: How to use it?

- Example: need to submit login information securely; script to execute is login.py
- Secure invocation:

Digital Certificates

- Bob got a public key from Amazon. Is it really Amazon’s key?
  - Trusted Certificate Authorities “sign” a certificate
  - Amazon’s certificate:
    - <Verisign, Amazon, https://www.amazon.com, Amazon’s public key>
    - Encrypted using Verisign’s private key
    - Browsers store Verisign’s public key, use it to decrypt the certificate, so they