MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Scripting for Cybersecurity

Interacting with Web Services

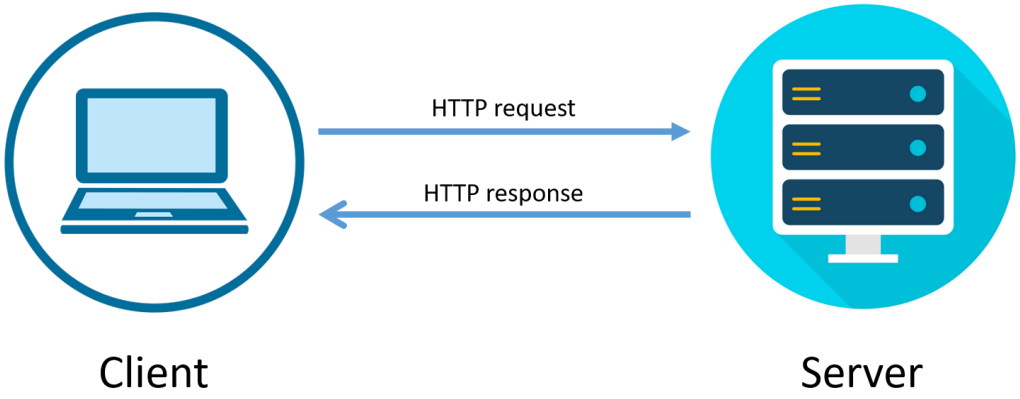
Dylan Smyth

- HTTP & Web Servers
- Linux Commands (wget & curl)
- Python Requests Library

Hypertext Transfer Protocol

- Hypertext Transfer Protocol (HTTP) is an application layer protocol used primarily to transport web pages and documents through the web

- When connecting to a website your web browser sends an HTTP request to a server which responds with the web page



https://bytesofgigabytes.com/IMAGES/Networking/HTTPcommuncation/http%20communication.png

Webpages

- Your browser may make subsequent requests to the server to get additional files for the website

- A website is usually consists of:
  - HTML
  - CSS
  - JavaScript
  - Images

Webpages

- Hypertext Markup Language (HTML) defines the structure and content of a webpage
- Cascading Style Sheets (CSS) is used to define the design and aesthetics of a webpage
- JavaScript is used for dynamic content on a webpage

Webpages

- The CSS and JS for a page may be split into separate files
- The HTML of the webpage will link to these files
- Once your browser has all of the necessary files for a webpage it will render the page

HTML

- A simple HTML webpage would look like this

```
<html>
  <head>
    <h1>Title of Something</h1>
  </head>
  <body>
    <p> Some content </p>
  </body>
</html>
```

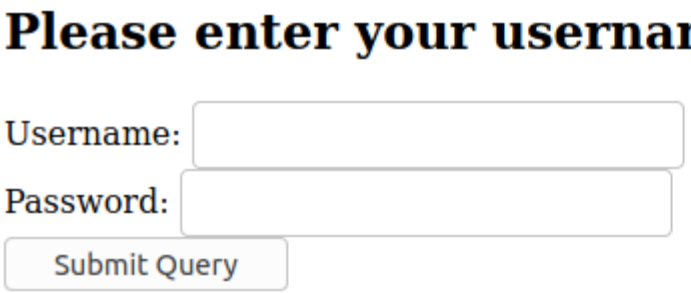
HTML

- The previous example would result in this:

HTML

- Forms allow users to submit data like usernames, passwords, files, search criteria, etc.

```
<form id='login_form' method='post' action='login.php'>
    <h2> Please enter your username and password </h2>
      Username: <input id='username' name='username' type='text'/></br>
      Password: <input id='password' name='password' type='password'/></br>
      <input type='submit'/>
  </form>
```



HTML

- When users submit data, some code running on the web server will do something with that data
- In the previous example, the data was sent to a page called “login.php”
- This data would be said to be handled *server-side*

Client-side vs Server-side

- HTML, CSS, and JavaScript are all rendered or executed on the client-side (i.e. in your browser)
- All of this code is accessible (right click -> view source)
- Server-side code on the other hand can be used to modify the HTML, CSS, and JavaScript before it's sent to the client
- Server-side code is also used to handle data submitted by a client (e.g. check if a username and password are correct)

Client-side vs Server-side

- Take the login.php file as an example:
  - The client sends a request for the login.php page
  - The server sends back the HTML containing a form that accepts a username and password
  - The user types in their username and password, then presses the submit button
  - The username and password is sent to the server. The server checks it
  - If the credentials are correct then then sends back HTML containing a welcome message.

Client-side vs Server-side

- A multitude of different server-side languages exist
- A large number of programming languages can be used as server-side languages for web servers. Some are more suited than others
- Some languages are built as a server-side language (PHP) whereas others have some version adapted for it (Java & JSP)

HTTP Requests

- As a protocol HTTP is very much request-response
- The client sends a request (for login.php for example)
- The server sends back a reply
- The clients request may contain data
- The server response may be X or Y depending on data in the clients request

HTTP Requests

- HTTP has a set of defined *Request Methods*
- The two most common methods are GET and POST
- Other types, like PUT and DELETE, are generally only seen when dealing with Application Programming Interfaces (APIs) based on HTTP (e.g. REST)

HTTP GET

- A HTTP GET request is used when a client wants to *get* a webpage

- When you connect to a website your browser will send a GET request for the main webpage and may send other GET request to fetch JavaScript, CSS, and image files for the webpage.

- A GET request is sent for a specific URL

URLs

- Uniform Resource Locator (URL) is essentially an address for something on the web
  - https://google.com
  - http://test.com/index.html
  - https://cs.cit.ie/
  - https://cs.cit.ie/contentfiles/PDFs/MScFlyers/MScCyber-online.pdf

- URLs are structured in a certain way…

URLs

- https://www.cit.ie/index.html

- **https://** -> Protocol
- **www** -> Sub-domain
- **cit.ie** -> Domain
- **/index.html** -> Requested file

- If no file is specified then the index page will be served by the server

URLs

- A URL can contain some user supplied data

https://example.com/search.php?query=cyberia

- In the above, a parameter *query* is passed into the search.php file with a value of *cyberia*

- The search.php file would then read in this value and perform some action with it

URLs

- Several *GET Parameters* can be provided:

https://example.com/search.php?query=cyberia**&x=1&y=2**

- The collection of parameters at the end of a URL is sometimes referred to as a *query string*

HTTP GET

- When a client wants a webpage from a server it will establish a TCP connection with the server and then send an HTTP GET request for that page

- A HTTP GET request looks like this:

```
GET /index.php HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

HTTP GET

```
GET /index.php HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

- The above example shows a typical GET request
- The file we're requesting is included, as well as our user-agent (identifying our browser), and more

HTTP GET

- A response from a server for a GET request looks like this:

```
HTTP/1.1 200 OK
Host: 127.0.0.1:8080
Date: Sun, 29 Nov 2020 19:36:19 GMT
Connection: close
X-Powered-By: PHP/7.4.3
Content-type: text/html; charset=UTF-8
```

- The data (the webage) is also included in the response

```
▸ Hypertext Transfer Protocol
▾ Line-based text data: text/html (8 lines)
    \n
    <html>\n
    <title>Index</title>\n
    <head></head>\n
    <body>\n
    <a href='/login.php'>Please log in to your account here</a>\n
    </body>\n
    </html>\n
```

HTTP GET

- Parameters in URLs are usually used to help navigate around the site

- User supplied details, such as a name, address, password, etc are usually supplied via a POST request instead

HTTP POST

- A POST request is similar to a GET request, except that data is provided in the HTTP header instead of using URL parameters

- POST requests are usually used when a user wants to supply personal information to the site (i.e. login credentials)

HTTP POST

- A POST request looks like this:

```
POST /login.php HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 45
Origin: http://127.0.0.1:8080
Connection: keep-alive
Referer: http://127.0.0.1:8080/login.php
Upgrade-Insecure-Requests: 1
```

- The data provided by the user is supplied along with this

```
▸ Hypertext Transfer Protocol
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "username" = "ubuntu"
  ▸ Form item: "password" = "thispasswordwillwork"
```

HTTP Response Codes

- In the case of both the GET and POST requests, the server sends a response status code
- This code is used to indicate the success or failure of the client's request, as well as why the request may have failed
- https://httpstatuses.com/

wget

- There are several tools available for the Linux command line to allow you to interact with anything available over the web

- One such tool is wget

```
ubuntu@ubuntu-VirtualBox:~/assignment_2$ wget
wget: missing URL
Usage: wget [OPTION]... [URL]...

Try `wget --help' for more options.
```

wget

- wget is used to download files from the web
- This can be used to download webpages, and clone entire websites
- It’s most commonly used to download files from a web server
- Can also be useful for transferring files between hosts

wget

```
ubuntu@ubuntu-VirtualBox:~/temp$ ls
ubuntu@ubuntu-VirtualBox:~/temp$ wget 127.0.0.1:8080/index.php
--2020-11-29 20:30:00--  http://127.0.0.1:8080/index.php
Connecting to 127.0.0.1:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘index.php’

index.php               [ <=>                ]     126  --.-KB/s    in 0s

2020-11-29 20:30:00 (21.1 MB/s) - ‘index.php’ saved [126]

ubuntu@ubuntu-VirtualBox:~/temp$ ls
index.php
ubuntu@ubuntu-VirtualBox:~/temp$
```

curl

- cURL is another tool that's useful for interacting with web services

- While wget is used to download files, curl offers more interaction for the user

```
ubuntu@ubuntu-VirtualBox:~/temp$ curl http://127.0.0.1:8080

<html>
<title>Index</title>
<head></head>
<body>
<a href='/login.php'>Please log in to your account here</a>
</body>
</html>
```

curl

- curl is a useful tool for interacting with websites without downloading any files from it

- You can create and execute more complex requests

```
ubuntu@ubuntu-VirtualBox:~/temp$ curl -X POST -d 'username=admin&password=thispasswordwillwork'
http://127.0.0.1:8080/login.php

<html>
<title>Login!</title>
<head></head>
<body><h1>Welcome admin!</h1></body>
</html>
```

Python Requests

- Python has a number of libraries available to allow integration with web services

- The Python Requests library is feature-rich and easy to use

```
>>> import requests
>>> resp = requests.get("http://127.0.0.1:8080/index.php")
>>> resp.status_code
200
>>> resp.text
"\n<html>\n<title>Index</title>\n<head></head>\n<body>\n<a href='/login.php'>Please log in to yo
ur account here</a>\n</body>\n</html>\n"
>>>
```

Python Requests

- The library can be imported using “import requests”

- An HTTP GET request can be sent like this:

```
resp = requests.get("http://127.0.0.1:8080/index.php")
```

- The data sent by the server can be accessed like this:

```
>>> print(resp.text)

<html>
<title>Index</title>
<head></head>
<body>
<a href='/login.php'>Please log in to your account here</a>
</body>
</html>
```

Python Requests

- A POST request can be sent be sent like so:

```
>>> data = {}
>>> data["username"] = "admin"
>>> data["password"] = "thispasswordwillwork"
>>>
>>> resp = requests.post("http://127.0.0.1:8080/login.php", data)
>>> resp.status_code
200
>>> print(resp.text)

<html>
<title>Login!</title>
<head></head>
<body><h1>Welcome admin!</h1></body>
</html>
```

- Note that the data posted by the client is in a dictionary

Python Requests

Demo!

MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Thank you