FANSHAWE

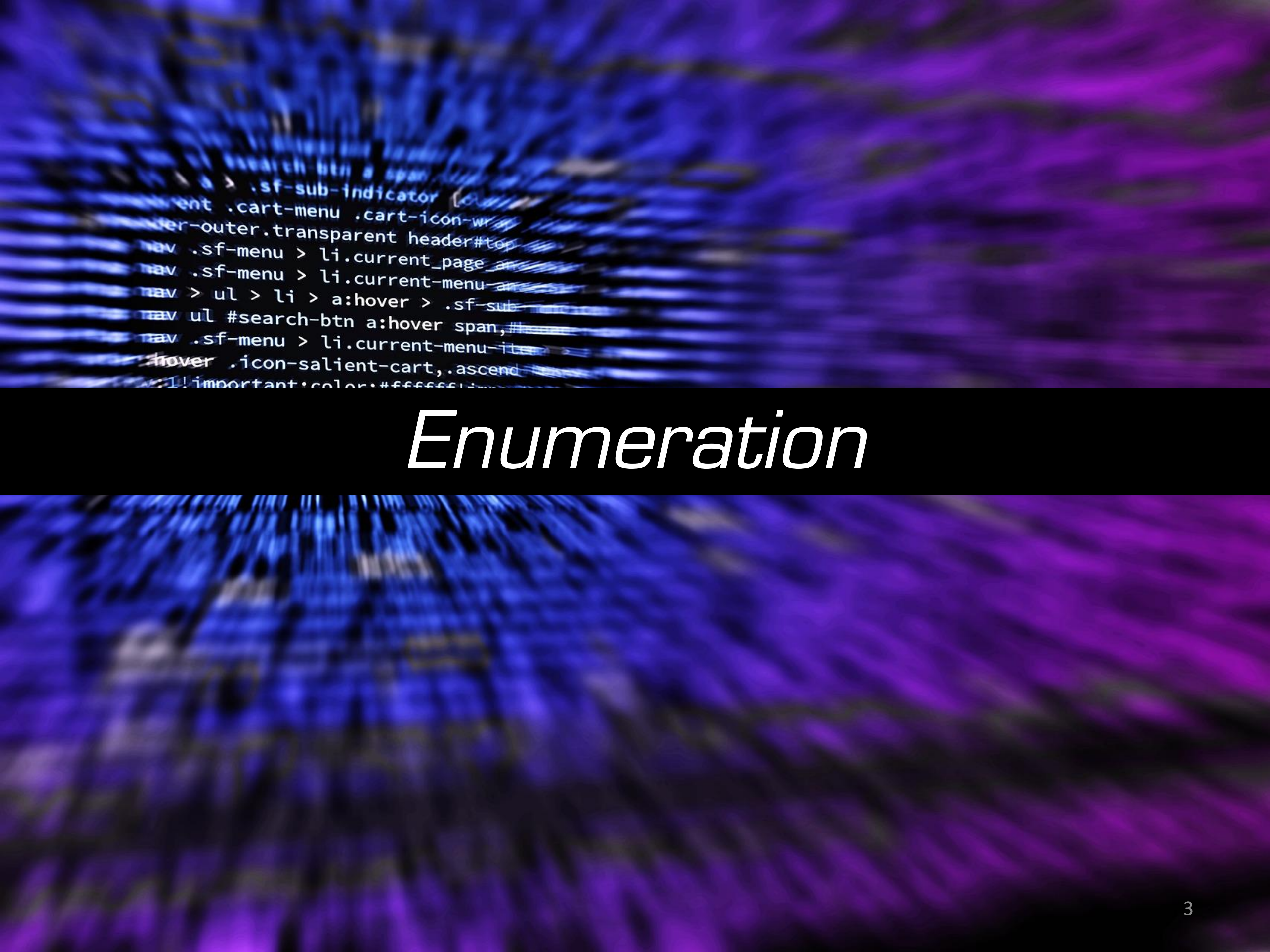
INFO-6076

Web Security

Web App Enumeration & HTTP

Agenda

- Enumeration
- Identifying technologies
- Mapping out the web application
- Spiders
- Discovering Content
- Discovering Functions
- Brute Force Attacks
- HTTP
- Lab 05 Overview



Enumeration

Enumeration

- Enumeration can be performed in two main ways
  - Web servers
  - Web applications
- Depending on the goal, one method may be beneficial over the other
- Most Web App penetration tests will involve just the Web App enumeration since it can be hosted on different platforms/servers

FANSHAWE

Server Enumeration

Server enumeration involves trying to find out what type of platform the web application is being hosted on

- What services are running on the server
- What O/S is the server running
- What versions of software are being used
- Etc.

Web App Enumeration

- Web App enumeration involves trying to find out what web technologies are being used
  - What functionality and what content is the application using?
  - How does the application behave?
  - What security mechanisms is it using?
  - Focuses on classes of vulnerabilities as opposed to instances

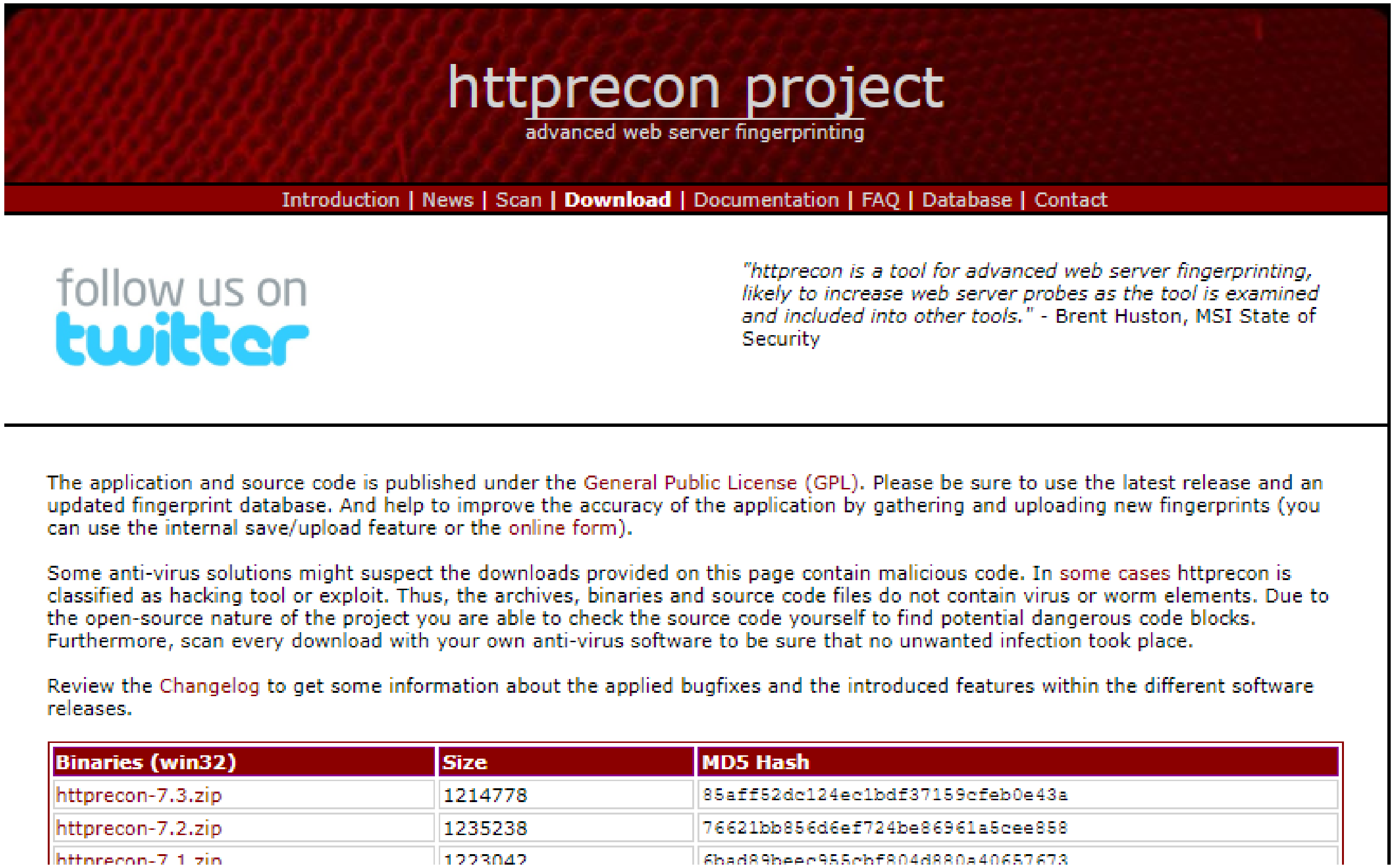
Identifying Technologies

Identifying Technologies

- Banner grabbing
  - HTTP Server Header
  - Templates used to build the HTML response
  - URL query string parameters
- The server banner can be changed by the administrator / developer
- There are tools available to attempt and discover the server technologies being used

Identifying Technologies

- HTTPrecon is a tool that does fingerprinting


httprecon project
advanced web server fingerprinting
Introduction | News | Scan | Download | Documentation | FAQ | Database | Contact
follow us on twitter

"httprecon is a tool for advanced web server fingerprinting, likely to increase web server probes as the tool is examined and included into other tools." - Brent Huston, MSI State of Security

The application and source code is published under the General Public License (GPL). Please be sure to use the latest release and an updated fingerprint database. And help to improve the accuracy of the application by gathering and uploading new fingerprints (you can use the internal save/upload feature or the online form).

Some anti-virus solutions might suspect the downloads provided on this page contain malicious code. In some cases httprecon is classified as hacking tool or exploit. Thus, the archives, binaries and source code files do not contain virus or worm elements. Due to the open-source nature of the project you are able to check the source code yourself to find potential dangerous code blocks. Furthermore, scan every download with your own anti-virus software to be sure that no unwanted infection took place.

Review the Changelog to get some information about the applied bugfixes and the introduced features within the different software releases.

Binaries (win32)	Size	MD5 Hash
httprecon-7.3.zip	1214778	85aff52dc124ec1bdf37159cfeb0e43a
httprecon-7.2.zip	1235238	76621bb856d6ef724be86961a5cee858
httprecon-7.1.zip	1223042	6bad89beec955cbf804d880a40657673

Identifying Technologies

File extensions also offer insight into what technologies are being used

- ASP – MS Active Server Pages
- PHP – PHP
- PY – Python
- JSP – Java Server Pages
- PL – Perl
- DLL – C or C++
- Etc.

Identifying Technologies

Fake file extensions can be used to observe the server's response in an attempt to discover what technologies are used when extensions are not normally shown

Examples:

- /view/fakepage.php
- /view/fakepage.asp

The server may respond with a customised message depending on the extension provided

Identifying Technologies

Other methods of identifying technologies include:

Directory names

- Servlet – Java Servlets
- Rails – Ruby on Rails

Session Tokens

- JSESSIONID – Java Platform
- PHPSESSID – PHP

Identifying Technologies

Third party components are often used by developers to speed up productivity

These can include things such as:

- Content Management Systems (CMS)
- Forums
- Shopping Carts
- Chat Bots
- Etc.

Mapping Out Web Applications

Mapping Out Web Applications

This stage involves mapping out the attack surface of a web application

Things to consider:

- URL strings and parameters used
- Parameters used with the POST method
- Cookies
- Session handling
- HTTP Headers

FANSHAWE

Mapping Out Web Applications

If an application is using REST-style URLs, it may contain parameters in the URL

- These can be tested as if they were queries

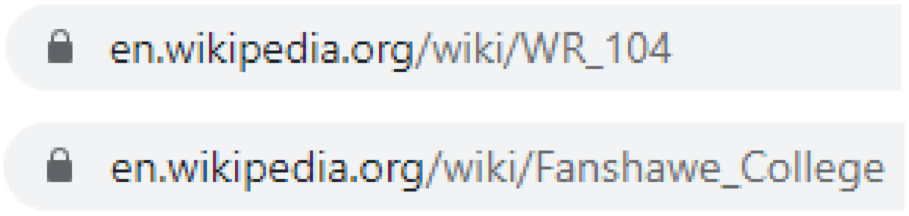
Example:

- http://artspizza/order/large/3topping

This URL can use **large** and **3topping** as parameters

Mapping Out Web Applications

These parameters can be changed and there is no pre-defined standard on how these are structured so manual testing may have to be performed


en.wikipedia.org/wiki/WR_104

en.wikipedia.org/wiki/Fanshawe_College

Mapping out the web application will help when it comes to focusing on potential vulnerabilities within the application

Spiders

Spiders

Spiders can be used to assist in mapping out a web application

- Spiders attempt to map an application out by going through web pages and following any links found
- Can also go through forms, entering specified or random data
  - Pull down menus, etc.

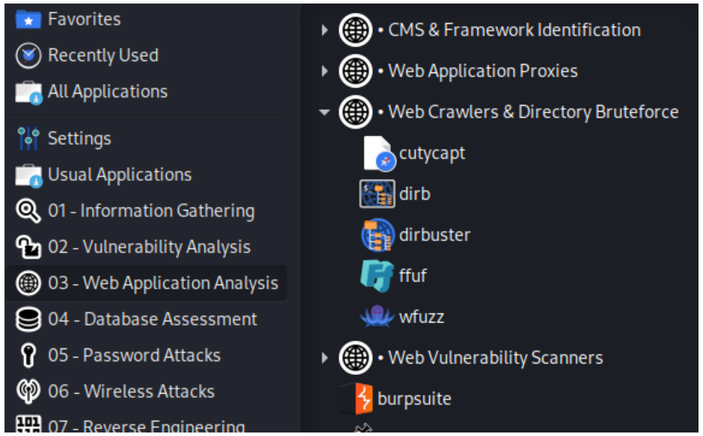
Spiders

Web servers may include a **robots.txt** file to tell web spiders which URLs they do not want indexed

- This file contains URLs that attackers will follow

Kali Linux contains several tools pre-installed for web app enumeration

- Burp Suite
- dirb
- Dirbuster


Favorites
Recently Used
All Applications
Settings
Usual Applications
01 - Information Gathering
02 - Vulnerability Analysis
03 - Web Application Analysis
04 - Database Assessment
05 - Password Attacks
06 - Wireless Attacks
07 - Reverse Engineering
• CMS & Framework Identification
• Web Application Proxies
• Web Crawlers & Directory Bruteforce
cutycapt
dirb
dirbuster
ffuf
wfuzz
• Web Vulnerability Scanners
burpsuite

Spiders

Automated spiders can have drawbacks

- They may not be able to navigate every link (if they cannot fill out the required information to move forward)
- They work on URLs
  - May miss data found at the same URL
  - May keep going if the URL is randomly created

Spiders

- Exercise caution when using spiders to map out a web application
- Can parse links to unprotected critical functions and begin sending data to them
- Could potentially:
  - Bring an application offline
  - Delete valuable data
  - Deface a web site

Spiders

- A proper test will involve manual user spidering in combination with a tool such as Burp Suite or WebScarab in order to map out an application with content and functionality
  - User controls the data being submitted in forms
  - Authentication can take place and links like **logout** can be avoided

Discovering Content

Discovering Content

- Web developers may keep content that is either obsolete, in testing, or a back up of the live web application
- Sometimes these are kept in the root folder of the live web application
- These will include the content and functionality that is not currently “live”

Discovering Content

- This **hidden** content may include things like:
  - Back up copies of files
  - Back up archives of the application
  - Default 3rd party code (WordPress, Drupal, etc.)
  - Old versions with vulnerabilities
  - Configuration and include files
  - Comment files
  - Log files

Discovering Functions

Discovering Functions

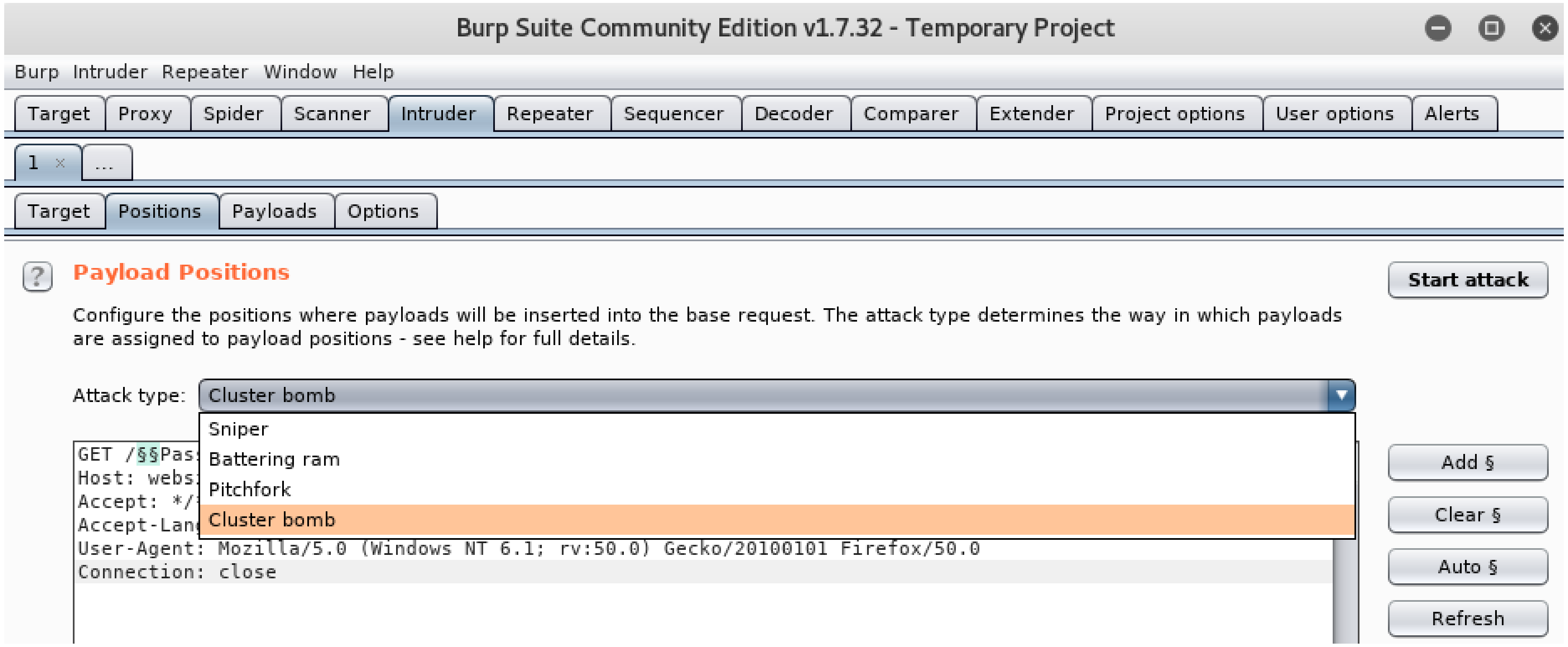
- Enumerating dynamic web content is a little different than static pages
- Requests to the server may all point to the same URL but have different functions
- Examples:
  - /orders.php?page=login
  - /orders.php?page=logout
  - /orders.jsp?action=placeorder
  - /orders.jsp?action=additem

Discovering Functions

- Sometimes developers add functionality that is not displayed by default
- You can test these parameters by using a wordlist for the parameter names and values
- Examples:
  - Debug=true
  - Test=true
- Burp Intruder has options for this purpose such as cluster bomb

Discovering Functions

- Burp Suite Intruder options:



Burp Suite Community Edition v1.7.32 - Temporary Project

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
1 × ...
Target Positions Payloads Options
Start attack
Add §
Clear §
Auto §
Refresh

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Cluster bomb

- Sniper
- Battering ram
- Pitchfork
- Cluster bomb

```
GET /§§Pas
Host: webs
Accept: */
Accept-Lan
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:50.0) Gecko/20100101 Firefox/50.0
Connection: close
```

FANSHAWE

Discovering Functions

- Test both the GET and POST methods
- For GET methods, test the parameters in the URL
- For POST methods, test the parameters included in the body of the message
- Focus on the functions where debugging would have been used by the developers

Brute Force Attacks

Web App Penetration Testing

Ideally, penetration testing will be done on a test environment/server so that if anything goes wrong, it does not affect the production machine and/or the web application

- Database content can be replaced with dummy data so that private information is not disclosed to the testers
- A duplicate environment is ideal

Brute Force Attacks

- Once you have the web application mapped out, you can use tools such as Burp Suite’s Intruder function to enumerate common directories such as:
  - About Us
  - Accounts
  - Images
  - Contact Us
  - Etc.

Brute Force Attack methods

- Check the results from the spider for valid directories or paths on the server
- Send requests for valid and invalid resources and note down the responses from the server
- Create a wordlist of common files and directories and use them against the discovered directories on the server
- Manually review the responses for valid resources

Brute Force Attack methods

- Check to see if there are any resources available in directories where authentication is required
- Note down the way the web application treats error responses
  - 401 Unauthorized
  - 403 Forbidden
  - 404 Not found
  - 500 Internal Server Error
  - Etc.

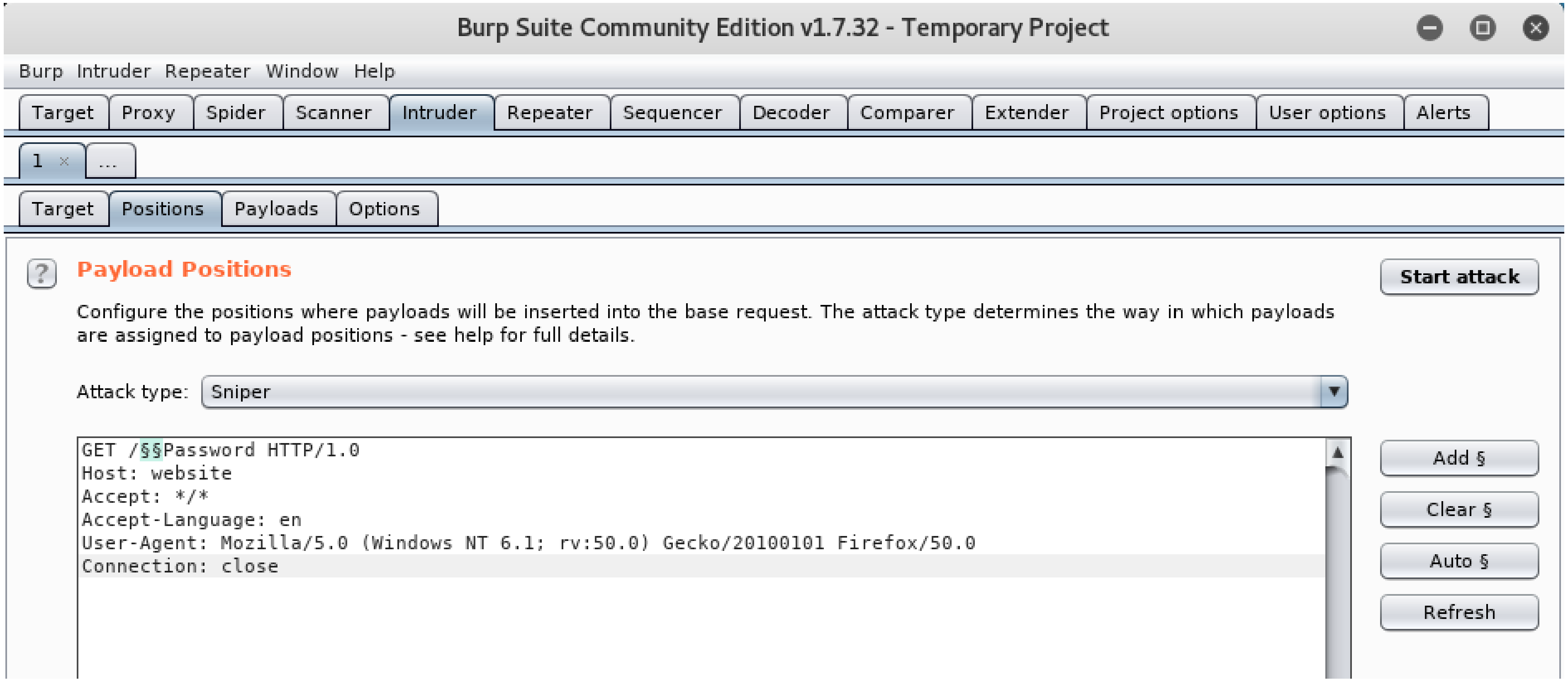
FANSHAWE

Brute Force Attack methods

- Developers typically have a specific naming scheme and stick to it
  - For Example:
  - http://website/ForgotPassword.php
  - http://website/ResetPassword.php
  - http://website/UpdatePassword.php
- Based on this, you may use wildcards to test for similar pages such as **RetrievePassword.php**

Brute Force Attack methods

- Burp Intruder is customizable as shown below:


Burp Suite Community Edition v1.7.32 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
1 × ...
Target Positions Payloads Options
Start attack
Add §
Clear §
Auto §
Refresh

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
GET /§§Password HTTP/1.0
Host: website
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:50.0) Gecko/20100101 Firefox/50.0
Connection: close
```

Brute Force Attack methods

Search for temporary files and source files depending on the technology used to create the web application

Examples of extensions to check for:

- .DS_Store
- .php~1
- .tmp
- .old

Brute Force Attack methods

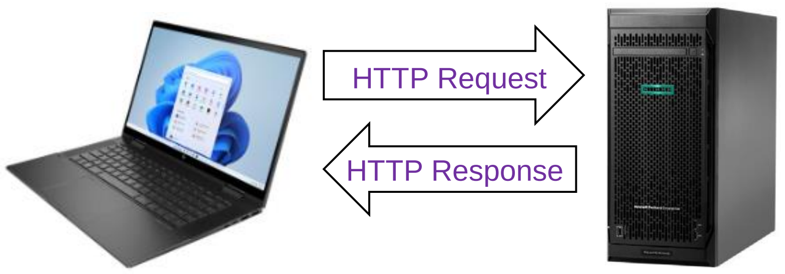
Searching the web application through Google may reveal some interesting information that can be used to attack it

- Cached content and functionality
- Groups and news
- Specific Queries:
    - Site: www.example.com
    - Link: www.example.com
    - Related: www.example.com

HTTP PROTOCOL

HTTP: HyperText Transfer Protocol

A set of rules for exchanging files such as text, graphic images, sound, video, and other multimedia files on the Web



Client-Server Model

Web browsers send HTTP requests for web pages and any associated files

Web servers send HTTP responses back to the web browsers

HTTP is a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it. It was originally designed with the sole purpose of retrieving static HTML pages

This is the <u>main reason</u> that it is difficult to implement Web sites that react intelligently to user input

This shortcoming of HTTP is being addressed in several technologies, including ActiveX, Java, JavaScript and cookies

HTTP: HyperText Transfer Protocol

Initially created by Tim Berners Lee at CERN

The standards development of HTTP was coordinated by the **Internet Engineering Task Force** (IETF) and the **World Wide Web Consortium** (W3C), culminating in the publication of a series of **Requests for Comments** (RFCs)

FANSHAWE

HTTP: HyperText Transfer Protocol

The following make up HTTP:

- Tim Berners Lee (1991) defined **HTTP/0.9**

  https://www.w3.org/Protocols/HTTP/AsImplemented.html
- **RFC 1945** (1996) defined **HTTP/1.0**
- **RFC 2068** (Jan 1997) defined **HTTP/1.1**
- **RFC 2616** (June 1999)
- **RFC 7230** (2014)
- **RFC 7540** (May 2015) defined **HTTP/2**
- **RFC 9114** (Jun 2022) defined **HTTP/3**

HTTP/2

- Based on a project created by Google called SPDY
    - SPDY has now been deprecated
- Compresses headers to increase transfer speed
- Compatible with HTTP/1.1
- Main goal is speed

https://en.wikipedia.org/wiki/HTTP/2

HTTP/3

- Most recent standard
- Based on a project created by the QUIC group
  - Originally called HTTP-over-QUIC
- Encodes headers and maintains session states
- Loads up to 3x faster than HTTP/1.1

https://en.wikipedia.org/wiki/HTTP/3

FANSHAWE

HTTP: HyperText Transfer Protocol

```
Shell>telnet amackiewicz2 80
Trying 10.0.0.200...
Connected to amackiewicz2
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: nginx
Date: Thu, 28 Jul 2022 13:32:18 GMT
Content-Type: text/html; charset=utf-8
Connection: close
Vary: Accept-Encoding
Set-Cookie: 1782e19debbe47f01ac06f75c41faa1f=kd9etl9ptko7e25cdg5dgo8443; expires =Thu, 24-Jul-2014 16:32:18 GMT; path=/
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
Cache-Control: no-cache
Pragma: no-cache

Connection closed by foreign host.
```

HTTP connection to mackiewicz.com (port 80)

HTTP Request: **HEAD / HTTP/1.0**
HEAD Method (same as **GET** but without text body)

HTTP Response

http://en.wikipedia.org/wiki/List_of_HTTP_header_fields#Responses

FANSHAWE

HTTP: Parameters

HTTP Protocol Parameters:
http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html

URI = **Uniform Resource Identifier** known by different names:
WWW, URL (Uniform Resource Locator). Unique identifier for a resource.

http_URL = **http: // host** : port [abs_path [? query]]

As far as HTTP is concerned, Uniform Resource Identifiers are simply formatted strings which identify--via name, location, or any other characteristic--a resource.

protocol : // host / folder / folder / file

http://artmackiewicz.com/finished/pic/boredroom.jpg

More Examples:

http://www.cnn.com
http://www.cnn.com/search/?query=world_news
http://www.cnn.com:80/search/?query=usa

FANSHAWE

HTTP: Connections

HTTP Connections:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8

HTTP Requests

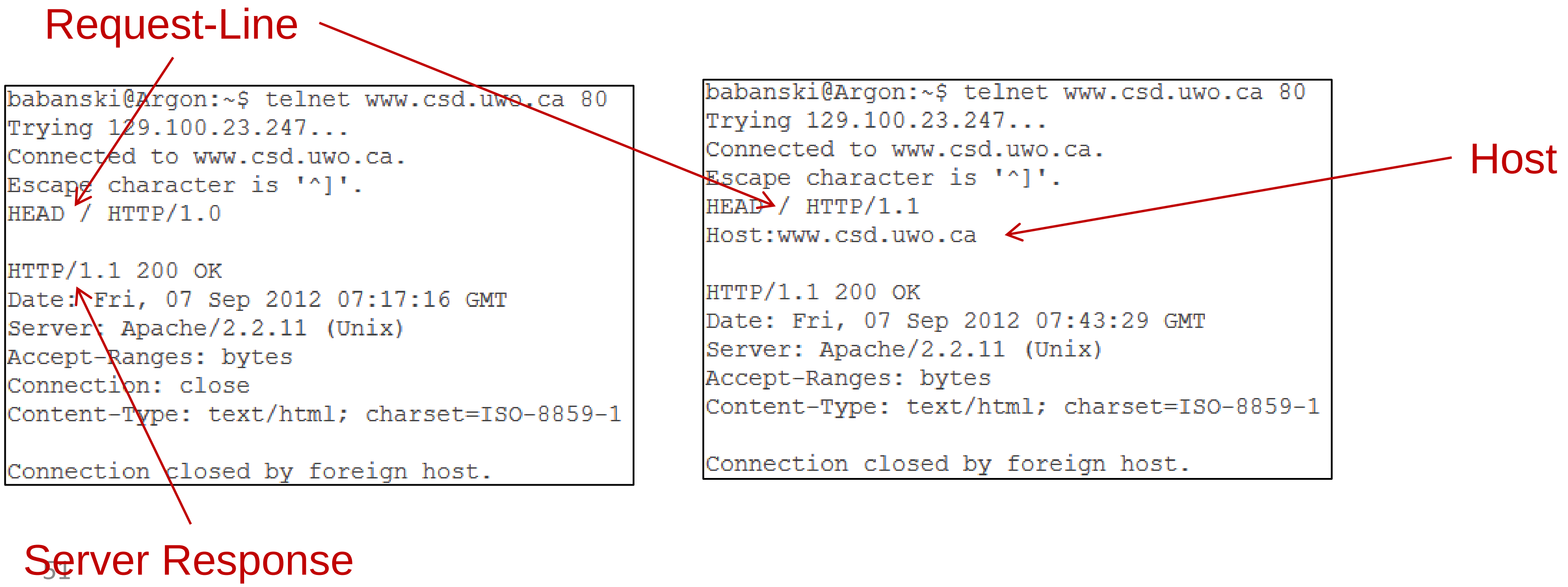
FANSHAWE

HTTP Requests

HTTP Requests:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.



```
babanski@Argon:~$ telnet www.csd.uwo.ca 80
Trying 129.100.23.247...
Connected to www.csd.uwo.ca.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Fri, 07 Sep 2012 07:17:16 GMT
Server: Apache/2.2.11 (Unix)
Accept-Ranges: bytes
Connection: close
Content-Type: text/html; charset=ISO-8859-1

Connection closed by foreign host.
```

```
babanski@Argon:~$ telnet www.csd.uwo.ca 80
Trying 129.100.23.247...
Connected to www.csd.uwo.ca.
Escape character is '^]'.
HEAD / HTTP/1.1
Host:www.csd.uwo.ca

HTTP/1.1 200 OK
Date: Fri, 07 Sep 2012 07:43:29 GMT
Server: Apache/2.2.11 (Unix)
Accept-Ranges: bytes
Content-Type: text/html; charset=ISO-8859-1

Connection closed by foreign host.
```

HTTP Requests

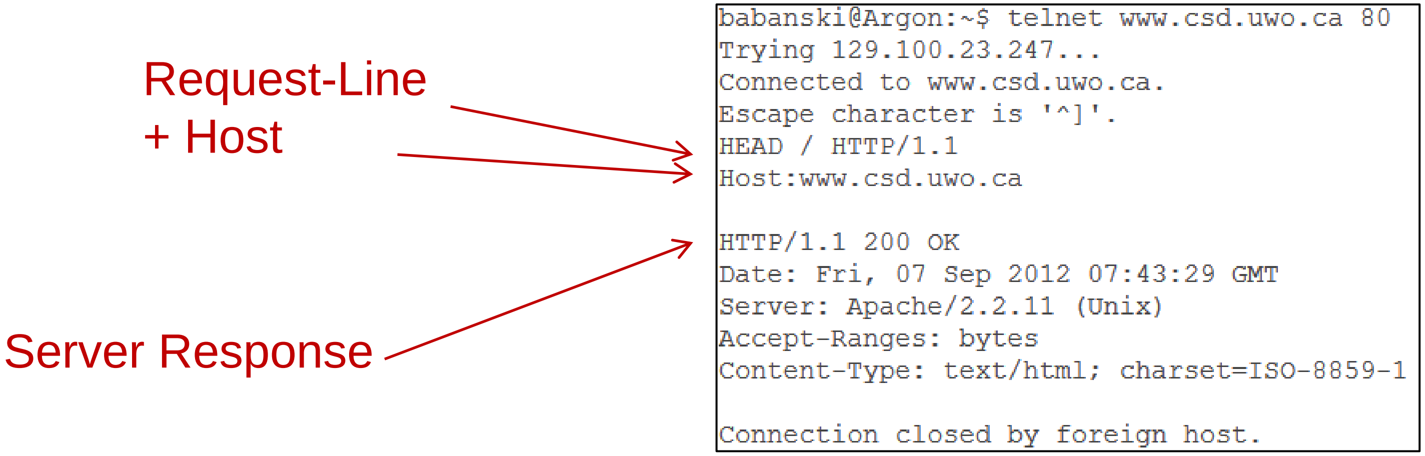
HTTP Requests:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5

The **Host** header is optional in **HTTP/1.0**, but it is mandatory in **HTTP/1.1**

The **Host** header distinguishes between various **DNS** names sharing a single **IP address**, allowing name-based **virtual hosting**.

Shared hosting companies:
BlueHost.com, GoDaddy.com, midphase.com, etc.



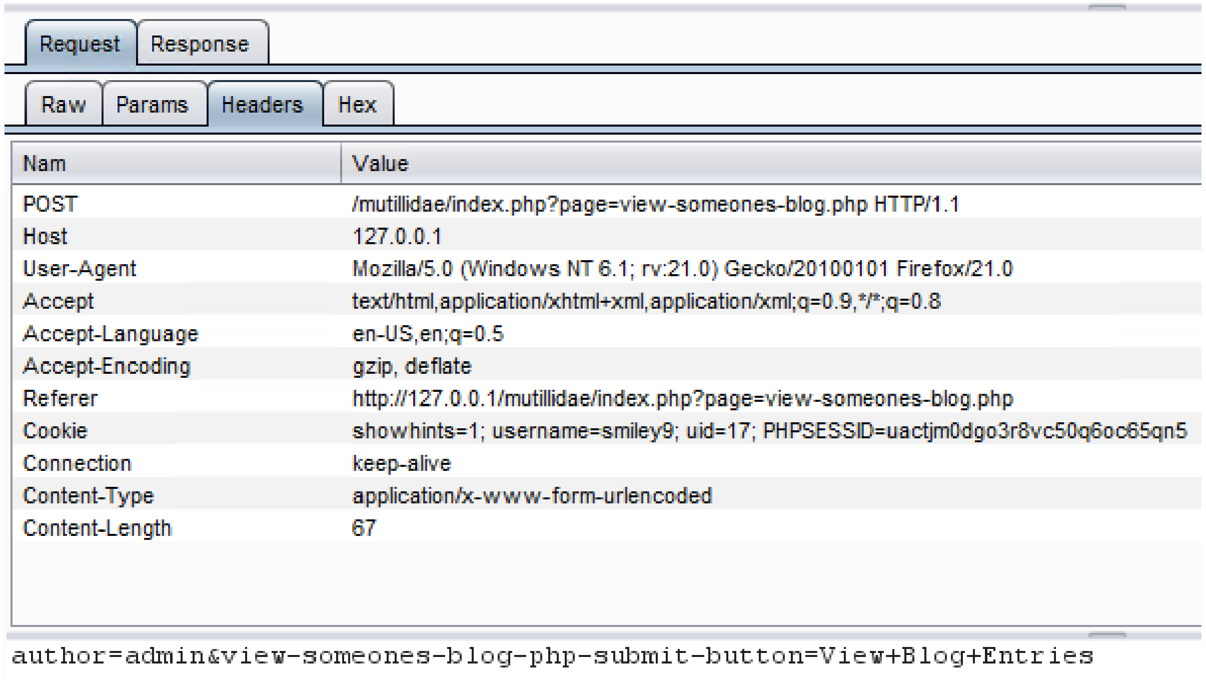
```
babanski@Argon:~$ telnet www.csd.uwo.ca 80
Trying 129.100.23.247...
Connected to www.csd.uwo.ca.
Escape character is '^]'.
HEAD / HTTP/1.1
Host:www.csd.uwo.ca

HTTP/1.1 200 OK
Date: Fri, 07 Sep 2012 07:43:29 GMT
Server: Apache/2.2.11 (Unix)
Accept-Ranges: bytes
Content-Type: text/html; charset=ISO-8859-1

Connection closed by foreign host.
```

FANSHAWE

HTTP Requests

Consist of one or more headers

- Each header is on a separate line
- Each header is separated by a blank line
- The headers are followed by an optional message body



Request | Response

Raw | Params | Headers | Hex

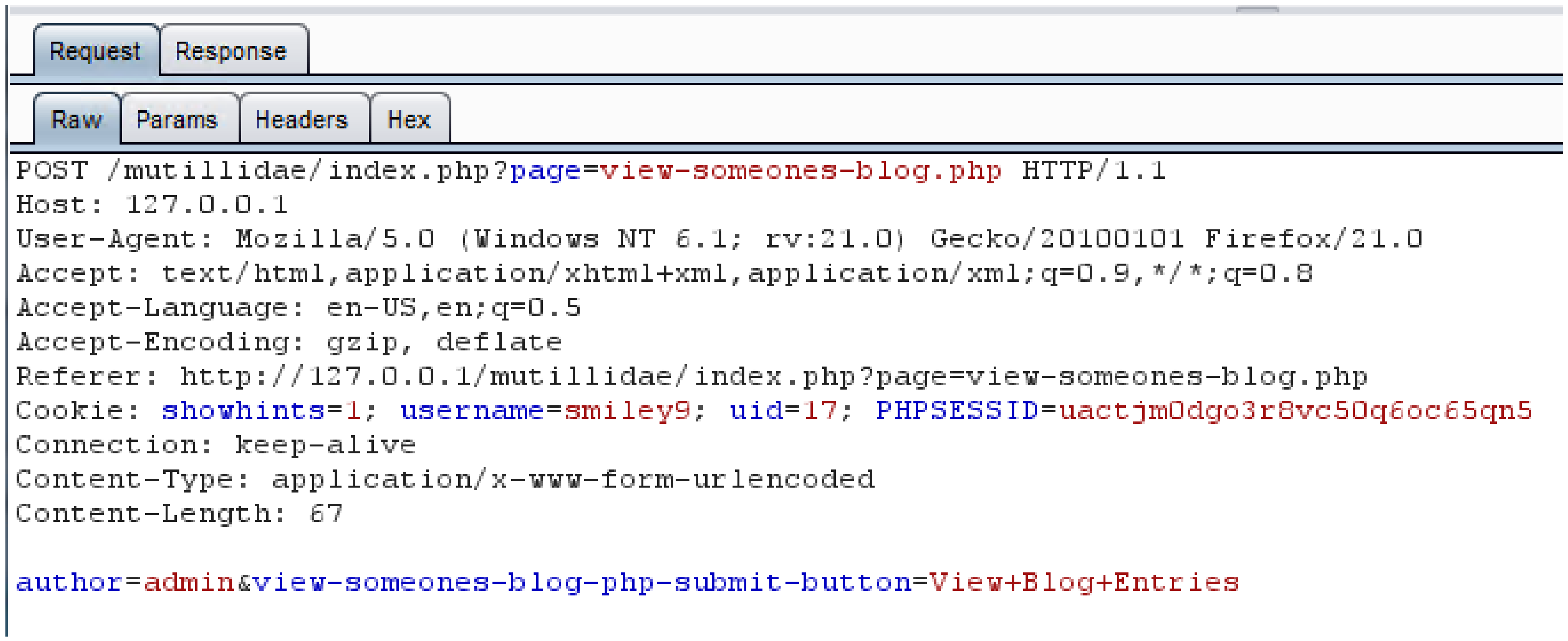
Nam	Value
POST	/mutillidae/index.php?page=view-someones-blog.php HTTP/1.1
Host	127.0.0.1
User-Agent	Mozilla/5.0 (Windows NT 6.1; rv:21.0) Gecko/20100101 Firefox/21.0
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
Referer	http://127.0.0.1/mutillidae/index.php?page=view-someones-blog.php
Cookie	showhints=1; username=smiley9; uid=17; PHPSESSID=uactjm0dgo3r8vc50q6oc65qn5
Connection	keep-alive
Content-Type	application/x-www-form-urlencoded
Content-Length	67

```
author=admin&view-someones-blog-php-submit-button=View+Blog+Entries
```

HTTP Requests

First line of every HTTP request consists of three items separated by spaces

- Method, URL and HTTP Version



```
POST /mutillidae/index.php?page=view-someones-blog.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:21.0) Gecko/20100101 Firefox/21.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/mutillidae/index.php?page=view-someones-blog.php
Cookie: showhints=1; username=smiley9; uid=17; PHPSESSID=uactjm0dgo3r8vc50q6oc65qn5
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 67

author=admin&view-someones-blog-php-submit-button=View+Blog+Entries
```

HTTP Requests

Method

- Indicates the HTTP method (GET, POST, HEAD, etc.)

URL

- This is the requested URL and is typically made up of the name of the resource and an optional query string

HTTP Version

- Identifies the HTTP version
  - 1.0 or 1.1 or 2.0
  - 2.0 being the current standard

HTTP Requests

The two of most interest are GET and POST

GET Method

- Designed to retrieve resources
- Can be used to send parameter in the URL query string

POST Method

- Designed to perform actions
- Parameters can be sent in the query string of the URL or the message body
- Generally, won't be reissued when user hits the back button
  - Prevents repeating actions

HTTP Requests

HEAD Method

- Can be used to check whether a resource is present before making a GET request
- Shouldn’t contain anything in the message body

TRACE Method

- Used for diagnostic purposes (what is received on the other end)
- Server should return the exact body contents from the message it received
  - Alterations can indicate changes made by a proxy

PUT Method

- Used to upload resources to the server
- Resources are contained in the body of the request

OPTIONS Method

- Requests a list of the HTTP methods supported by a resource

FANSHAWE

HTTP Requests

HTTP Requests Header Fields:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html#sec5.3
http://en.wikipedia.org/wiki/List_of_HTTP_header_fields

Important Fields

Accept

Content-Types that are acceptable.

Accept: text/plain

Accept-Charset

Character sets that are acceptable.

Accept-Charset: utf-8

Accept-Encoding

Acceptable encodings (HTTP Compression).

Accept-Encoding: gzip, deflate

Host

The requested domain name of the server (for virtual hosting)

FANSHAWE

HTTP Requests

Cache-Control
Allows a client or server to transmit a variety of directives, typically to override the default caching algorithms.
Freshness (**max-age** directive), Validation (**If-Modified-Since** directive).
Cache-Control: no-cache

Cookie
An HTTP cookie previously sent by the server with **Set-Cookie**.
Cookie: $Version=1; Skin=new;

Important Fields

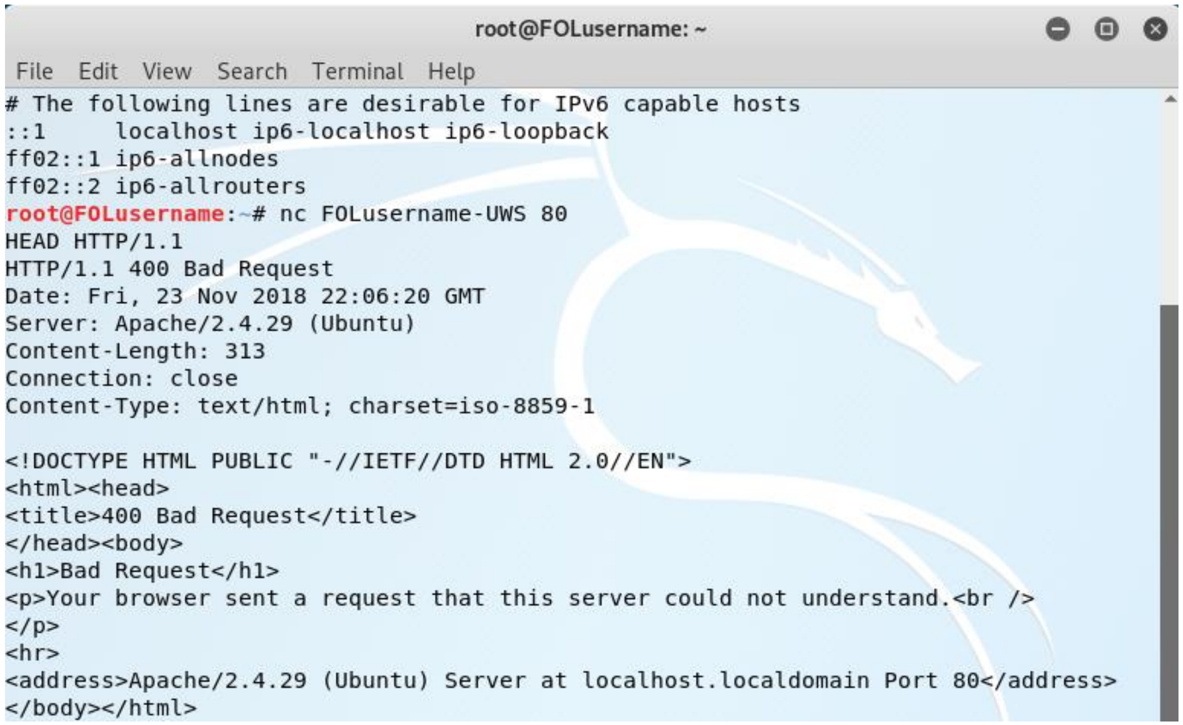
Referer (notice that it's not Referrer!)
The address of the previous web page from which a link to the currently requested page was followed.
Referer: http://en.wikipedia.org/wiki/Main_Page

User-Agent
The user agent string of the user agent.
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/12.0

HTTP Requests

NetCat

- Retrieve information from the web server
- HEAD HTTP/1.1
- Server O/S
- Apache Version



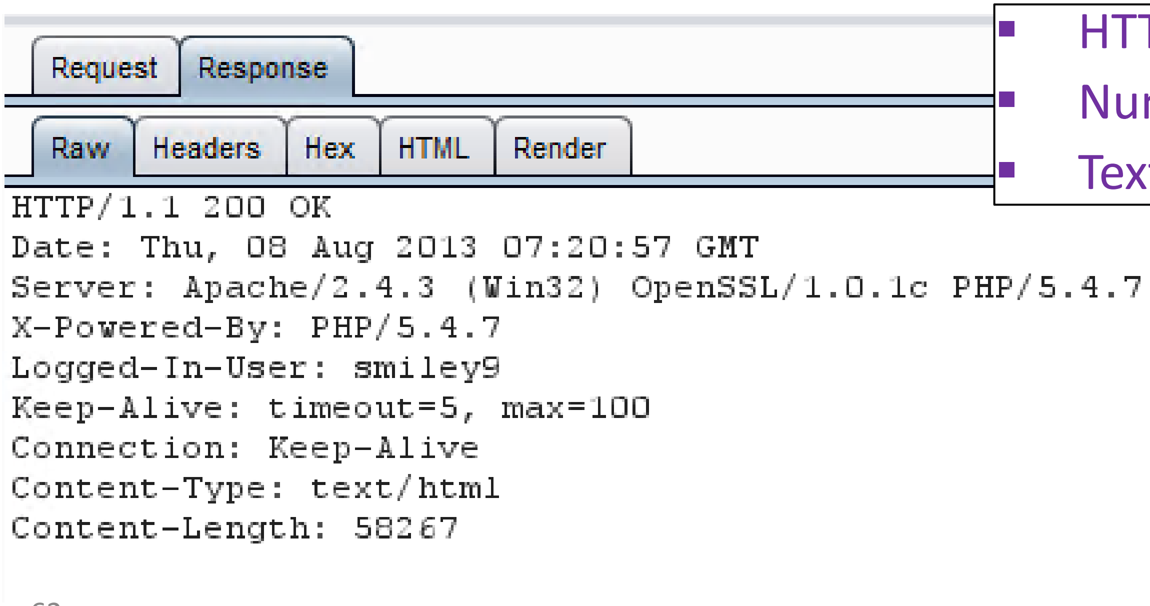
```
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
root@FOLusername:~# nc FOLusername-UWS 80
HEAD HTTP/1.1
HTTP/1.1 400 Bad Request
Date: Fri, 23 Nov 2018 22:06:20 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Length: 313
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.29 (Ubuntu) Server at localhost.localdomain Port 80</address>
</body></html>
```

HTTP Responses

HTTP Responses

As with the HTTP Request the first line consists of three items separated with spaces

- HTTP Version, Numeric Status Code, Textual Reason Phrase



```
HTTP/1.1 200 OK
Date: Thu, 08 Aug 2013 07:20:57 GMT
Server: Apache/2.4.3 (Win32) OpenSSL/1.0.1c PHP/5.4.7
X-Powered-By: PHP/5.4.7
Logged-In-User: smiley9
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
Content-Length: 58267
```

- HTTP Version: HTTP/1.1
- Numeric Status Code: 200
- Textual Reason Phrase: OK

FANSHAWE

HTTP Responses

HTTP Responses:
http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html

Status Codes:
http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

1xx: Informational (Request received, continuing process)
2xx: Success (The action was successfully received, understood, and accepted)
    200 OK
3xx: Redirection (Further action must be taken in order to complete the request)
    301 Moved Permanently
    302 Moved Temporarily (conflict with "Found" in browsers)
4xx: Client Error (The request contains bad syntax or cannot be fulfilled)
    403 Forbidden
    404 Not Found
5xx: Server Error (The server failed to fulfill an apparently valid request)
    500 Internal Server Error
    502 Bad Gateway

FANSHAWE

HTTP Responses

HTTP Response Header Fields:
http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6.2
http://en.wikipedia.org/wiki/List_of_HTTP_header_fields

Cache-Control
Tells all caching mechanisms from server to client whether they may cache this object.
Cache-Control: max-age=3600

Connection
Connections state.
Connection: close (vs Connection: keep-alive)

Important Fields

Location
Used in redirection, or when a new resource has been created.
Location: http://www.fanshaweonline.com

Content-Disposition
Opportunity to enforce "File Download" procedure. (RFC 1806 and RFC 2183)
Content-Disposition: attachment; filename="somefile.ext"

FANSHAWE

HTTP Responses

Content-Type
The MIME type of this content.
Content-Type: text/html; charset=utf-8

Expires
Gives the date/time after which the response is considered stale
Expires: Thu, 02 Feb 2023 16:00:00 GMT

Important Fields

Last-Modified
The last modified date for the requested object.
Last-Modified: Thu, 02 Feb 2023 12:45:26 GMT

Set-Cookie
Server requests that browser creates a cookie.
Set-Cookie: UserID=RyanGinger; Max-Age=3600; Version=1

FANSHAWE

HTTP Responses

Server Header

- Contains a banner indicating the web server software that is being used
  - Apache/2.4.3 (Win32)
- May also contain installed modules
  - OpenSSL/1.0.1c
  - PHP/5.4.7

Example: Server: Apache/2.4.3 (Win32) OpenSSL/1.0.1c PHP/5.4.7

Poses security threat! Frequently minimal or false information is given!

Example: Server: nginx

HTTP Responses

You can dynamically modify the **HTTP response header** to:

- Redirect the web client to another URL
- Send a different HTTP status code
- Tell the web client whether to cache the current document or not
- Tell the web client what language is used in the current document
- Change the content type of the current document
  - You can use PHP to dynamically create a text file, CSV file, image, etc.
- Requesting the web client to download another file
- Set cookies (but in PHP, cookies should be set through **$_COOKIE** instead)

Lab Overview

LAB-05: Overview

Lab-05: Web App Enumeration & HTTP

- Burp Suite Intruder
- Analyzing HTTP Requests and Responses
- Manipulating HTTP Request and Response information with Burp Suite
- Modifying HTTP User-Agent Header fields