Secure Web Development

By

Dr. Madani

Winter 2024

Learning Objectives

Upon completion of this material, you should be able to

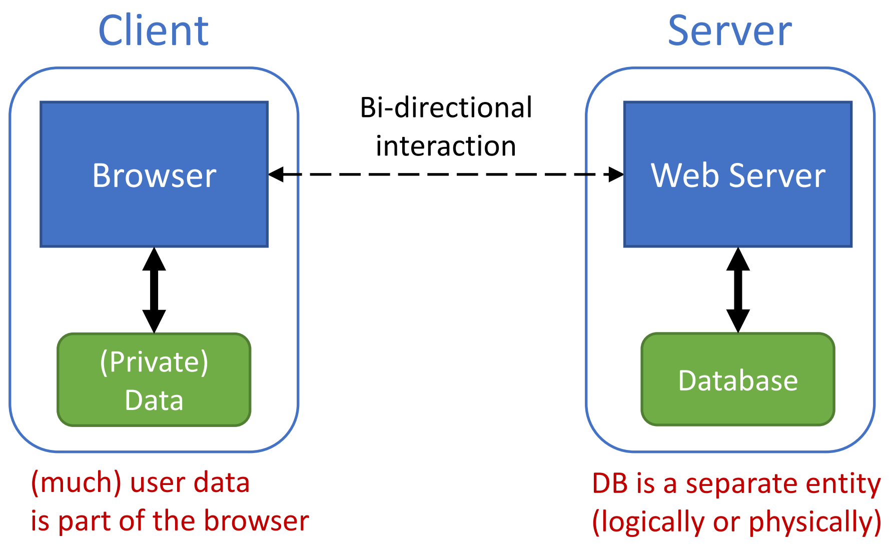
- Explain basics of HTTP protocol and WWW traffics.
- Understand the difference between GET and POST requests.
- Explain fundamental components of a web application such as Request/Response handling and Cookies and Sessions.
- Enumerate and describe different web-app specific attacks and defences.

Optional Reading

- Chapters 11 of Designing Secure Software by Kohnfelder

Web Basics

- The World Wide Web (www)
  - Roughly speaking, consist of two type of participants: (a) clients, and (b) servers that interact with on another.
  - Clients: e.g., your phone, laptops, desktops, etc., all of which are interested in content provided by servers.
    - Often runs a web browser (e.g., Internet Explorer, Edge, Chrome, Firefox, etc.): an application for accessing web contents served by the servers!
  - Servers: e.g., shopping website like Amazon, information web pages like Wikipedia.
    - Often runs a web server (e.g., Apache, HTTPD, EngX, MS-IIS) and database (e.g., MySQL, Postgresql, or MS-SQL) which keeps track of the information it's serving.


Client
Server
Browser
Bi-directional interaction
Web Server
(Private) Data
Database
(much) user data is part of the browser
DB is a separate entity (logically or physically)

Web Basics – cont'd

- Browser interaction with Web Servers
  - Uses Universal Resource Locators (URLs) to identify resources.
    - E.g., https://businessandit.ontariotechu.ca/index.php

Protocol
e.g., ftp
https, tor

A set of standards dictating how servers and clients must interact.

Hostname/Server Address

A human readable name that is translated into an internet address (i.e., IP) – 205.211.190.48 - by a Domain Name Service (DNS).
This translation is important to locate the server on the Internet.

Path to resource

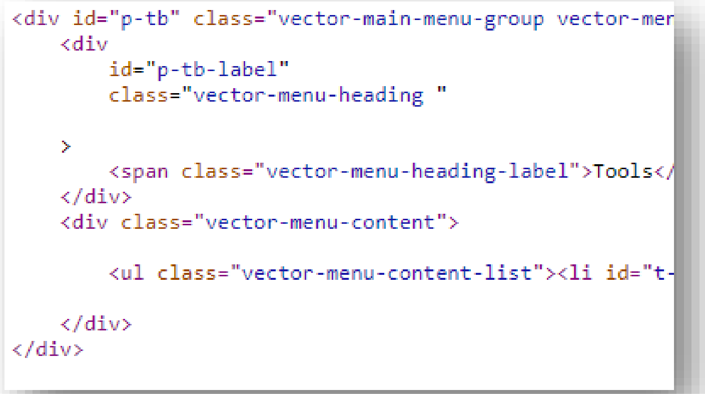
The remaining part of a URL is the path to the particular resource that the client is interested and server is providing!

Web Basics – cont’d

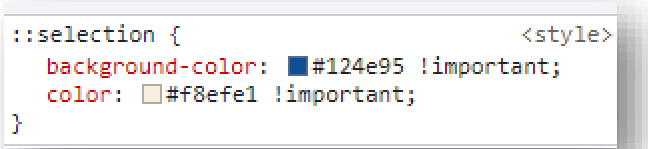
- Two general types of resources are served:
  - Static Contents
    - These are files stored on the server, and when requested, will be fetched and returned to the user.
    - E.g., Images, Video Files, Text Files, HTML Files, …
    - You can get a basic idea by paying attention to the file extension used in the resource path of the URL
      - https://businessandit.ontariotechu.ca/mytest.txt
  - Dynamic Contents
    - These contents are “dynamically” generated on-the-fly, when requested by the user. The requested resource may change over time and according to some environmental variables (e.g., time of the day, the requesting user, etc.)
    - Often, is generated as a result of running a “web program” developed using a particular programming language such as C/C++, C#-ASP.NET, PYTHON, PERL, PHP, …
      - https://businessandit.ontariotechu.ca/index.php
      - https://businessandit.ontariotechu.ca/JUSTTEST

Web Basics – cont’d

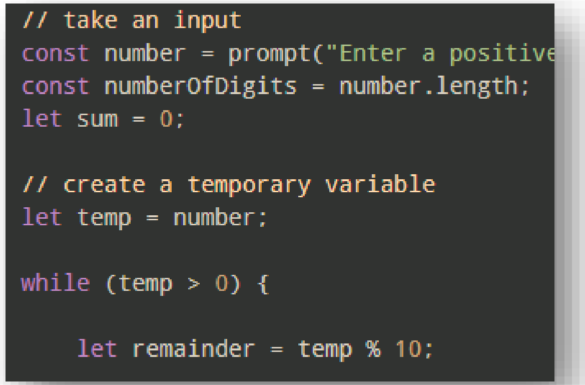
- HyperText Markup Language (HTML)
  - Is the standard language for documents designed to be displayed in a web browser. (a static content.)
  - It’s not a “programming” language, it is a markup language.
  - It is assisted by Cascading Style Sheets (CSS) and some scripting languages such as Java.
- Cascading Style Sheets (CSS)
  - A markup language for styling HTML pages.
- Client-side Scripting Language (e.g., JavaScript)
  - A client side program that will be executed on the client’s browser and can interact with HTML/CSS objects, client’s files (limited), and the Internet.



```
<div id="p-tb" class="vector-main-menu-group vector-mer
    <div
        id="p-tb-label"
        class="vector-menu-heading "

    >
        <span class="vector-menu-heading-label">Tools</
    </div>
    <div class="vector-menu-content">

        <ul class="vector-menu-content-list"><li id="t-

    </div>
</div>
```



```
::selection {                                   <style>
   background-color: ■#124e95 !important;
   color: □#f8efe1 !important;
}
```

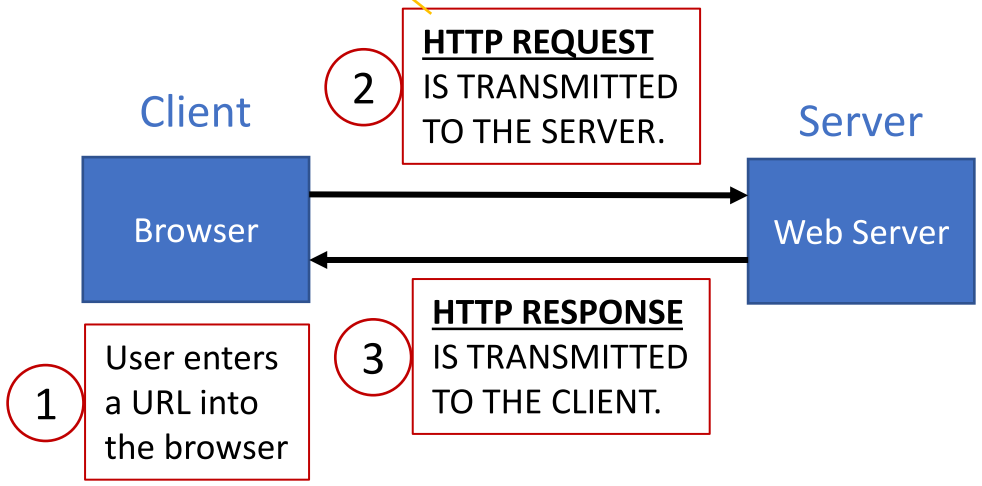


```
// take an input
const number = prompt("Enter a positive
const numberOfDigits = number.length;
let sum = 0;

// create a temporary variable
let temp = number;

while (temp > 0) {

    let remainder = temp % 10;
```

Web Basics – cont’d

- Basic structure of web traffic
  - The browser communicates with the web server using the protocol that’s specified in the URL.
  - The most common protocol (the one we’ll focus on) is HTTP and HTTPS.
    - Stands for HypterText Transfer Protocol (HTTP).
    - It’s a so-called application layer protocol when using the OSI network stack protocol and it runs on top of TCP.
    - TCP is used to exchange collections of data reliably across a network even if the network itself is unreliable!

Request contains
1. The URL of the resource clients wishes to obtain.
2. Headers describing what the browser can do and type of the request (GET vs POST).


Client
Server
Browser
Web Server
2
HTTP REQUEST IS TRANSMITTED TO THE SERVER.
1
User enters a URL into the browser
3
HTTP RESPONSE IS TRANSMITTED TO THE CLIENT.

Web Basics – cont’d

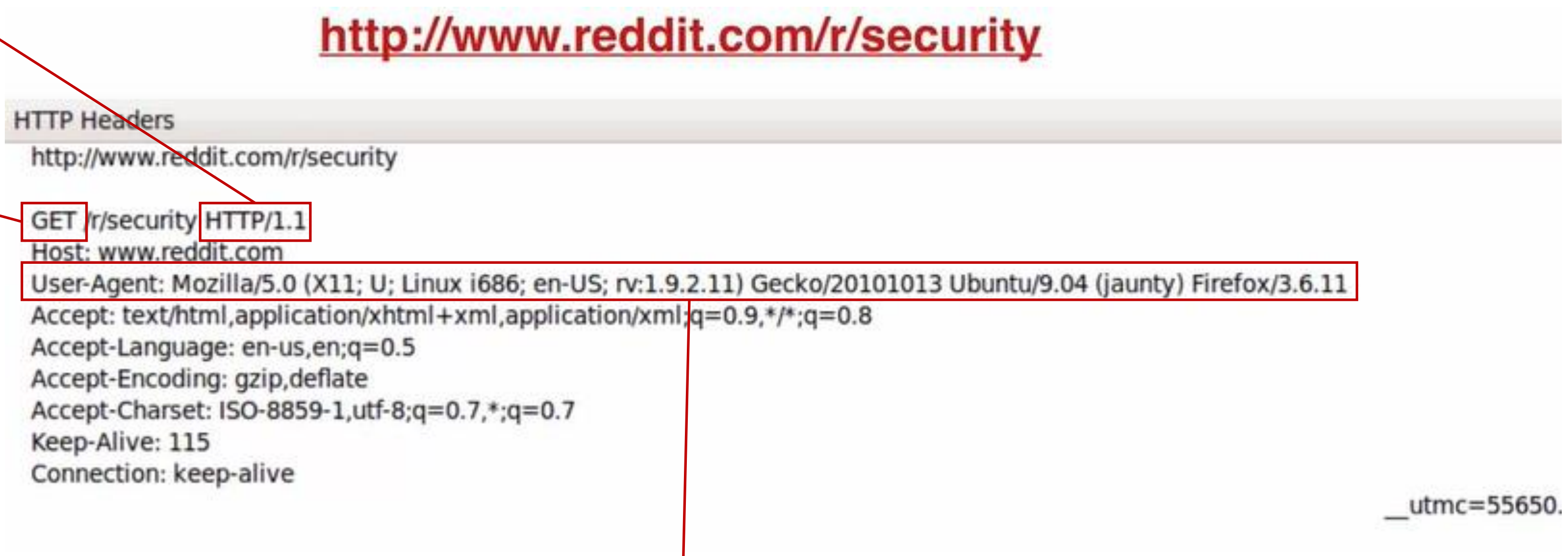
- Basic structure of web traffic – cont’d
  - The request:
    - Can be initiated by clicking on a link or typing a URL in the address bar of the browser.
    - Also it can be created by filling out and submit an HTML form.
  - Request types can be GET or POST
    - There are some other types that we are not concerned with in this course!
    - GET: all the necessary request data is in the URL itself and it generally should not have any server side effect.
      - That is to say, using this type of request, generally, the user request some resources from the server without asking for any modifications to the requested materials and resources.
      - E.g., http://www.ontariotechu.ca/todaynews.php?today=$\boxed{\text{2023-02-13}}$
    - POST: includes the data as separate fields in the request header and it generally may have some side effects on the data/resources maintained by the server.

Web Basics – cont’d

- An example of HTTP GET request

Protocol used and its version

Type of the request



http://www.reddit.com/r/security

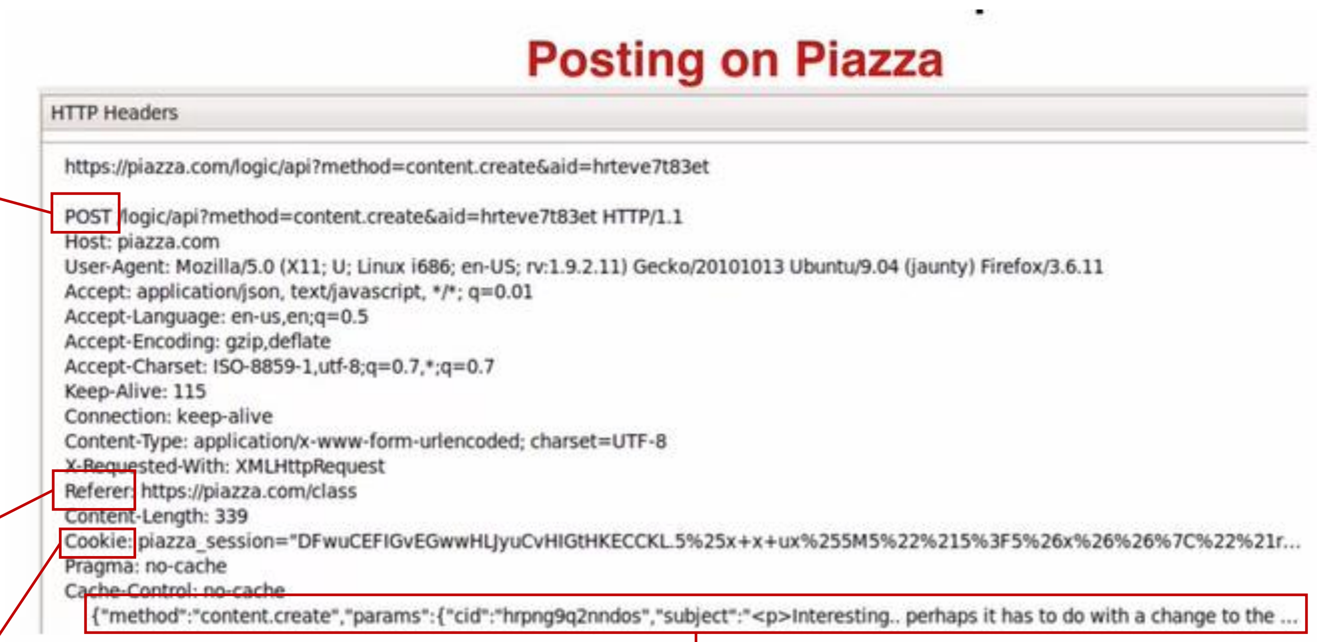
HTTP Headers

```
http://www.reddit.com/r/security

GET /r/security HTTP/1.1
Host: www.reddit.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
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
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
                                                                                           __utmc=55650.
```

User-agent is typically identifies the browser the client is using.

Web Basics – cont’d

- An example of a HTTP POST request



Posting on Piazza

HTTP Headers

```
https://piazza.com/logic/api?method=content.create&aid=hrteve7t83et

POST /logic/api?method=content.create&aid=hrteve7t83et HTTP/1.1
Host: piazza.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20101013 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: https://piazza.com/class
Content-Length: 339
Cookie: piazza_session="DFwuCEFIGvEGwwHLJyuCvHIGtHKECCKL.5%25x+x+ux%255M5%22%215%3F5%26x%26%26%7C%22%21r...
Pragma: no-cache
Cache-Control: no-cache
    {"method":"content.create","params":{"cid":"hrpng9q2nndos","subject":"<p>Interesting.. perhaps it has to do with a change to the ...
```

Type of the request

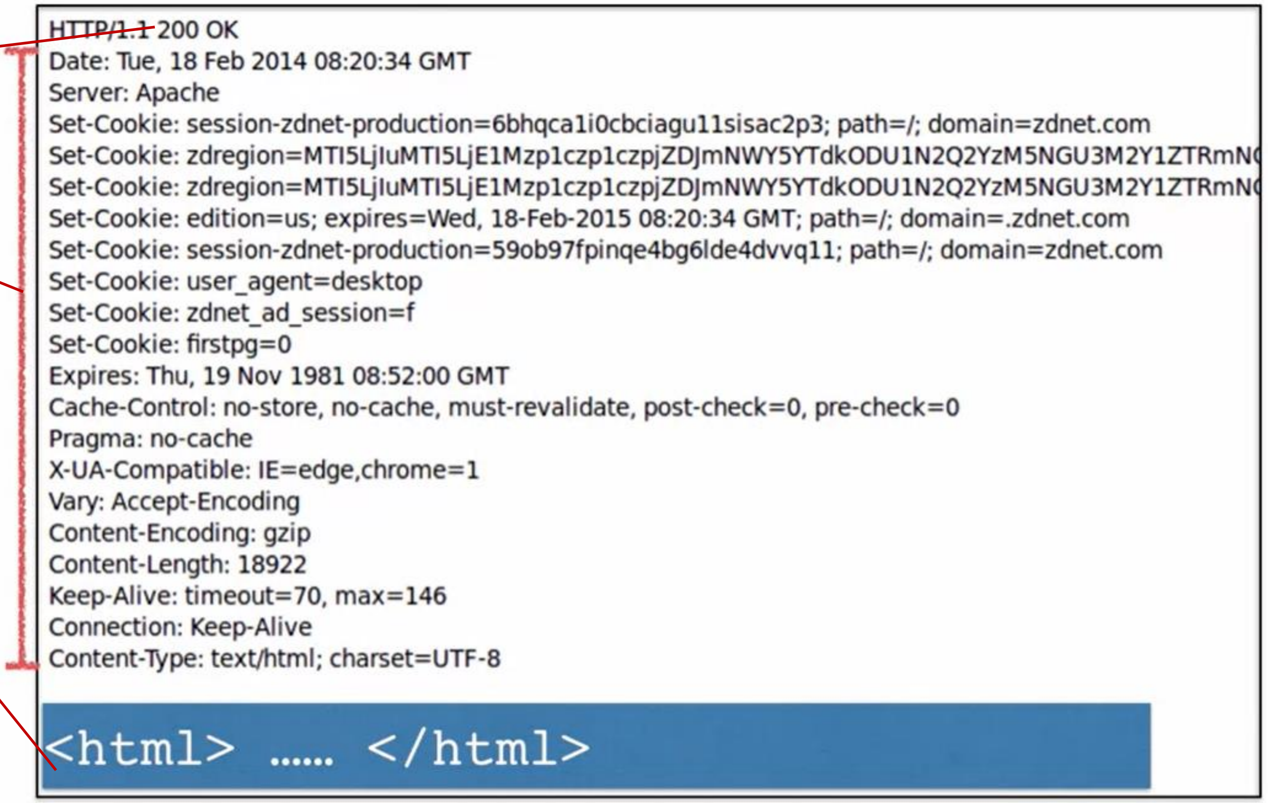
Referrer field identifies if where the user is coming from. Typing the URL in the address bar of a browser does not have any referrer.
However, a request generated by clicking on a link, has a referrer. When click on a link to a website on a Google search result page, the referrer will be set to Google.com!

We will talk about it soon!

The client is submitting some form data to the server to be processed.

Web Basics – cont’d

- Response received from the server:
  - A status code.
  - Headers describing what server provides.
    - Cookies – that represent some “state data” the server would like the browser to store on its behalf! (part of the header!)
  - Data – e.g., HTML, CSS, Image, …



```
HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: zdregion=MTI5LjIuMTI5LjE1Mzp1czp1czpjZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
Set-Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvvq11; path=/; domain=zdnet.com
Set-Cookie: user_agent=desktop
Set-Cookie: zdnet_ad_session=f
Set-Cookie: firstpg=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-UA-Compatible: IE=edge,chrome=1
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 18922
Keep-Alive: timeout=70, max=146
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<html> ...... </html>
```

HTTP Cookies

- Small block of **key/value** data created by a web server and stored on the user's device or web browser. E.g.,:
  - For storing stateful information (*such as items that you have added to your shopping cart!*).
  - Authenticated Session Tokens.
- Types of Cookies:
  - Session Cookie: exist only in temporary memory of the server, while the user is navigating the website. They expire or deleted when the user closes the web browser. Usually do not have expiration date assigned to them.
  - Persistent Cookie: expires at a specific date (decided by the web server). Often used for tracking.
  - Secure Cookie: are transmitted over an encrypted connection only (is a property that session or persistent cookies can have!).
  - Http-only Cookie: type of cookies that cannot be accessed by client-side APIs such as JavaScript. This eliminates cookie threats via cross-site-scripting (XSS) attack!

HTTP Cookies – cont'd

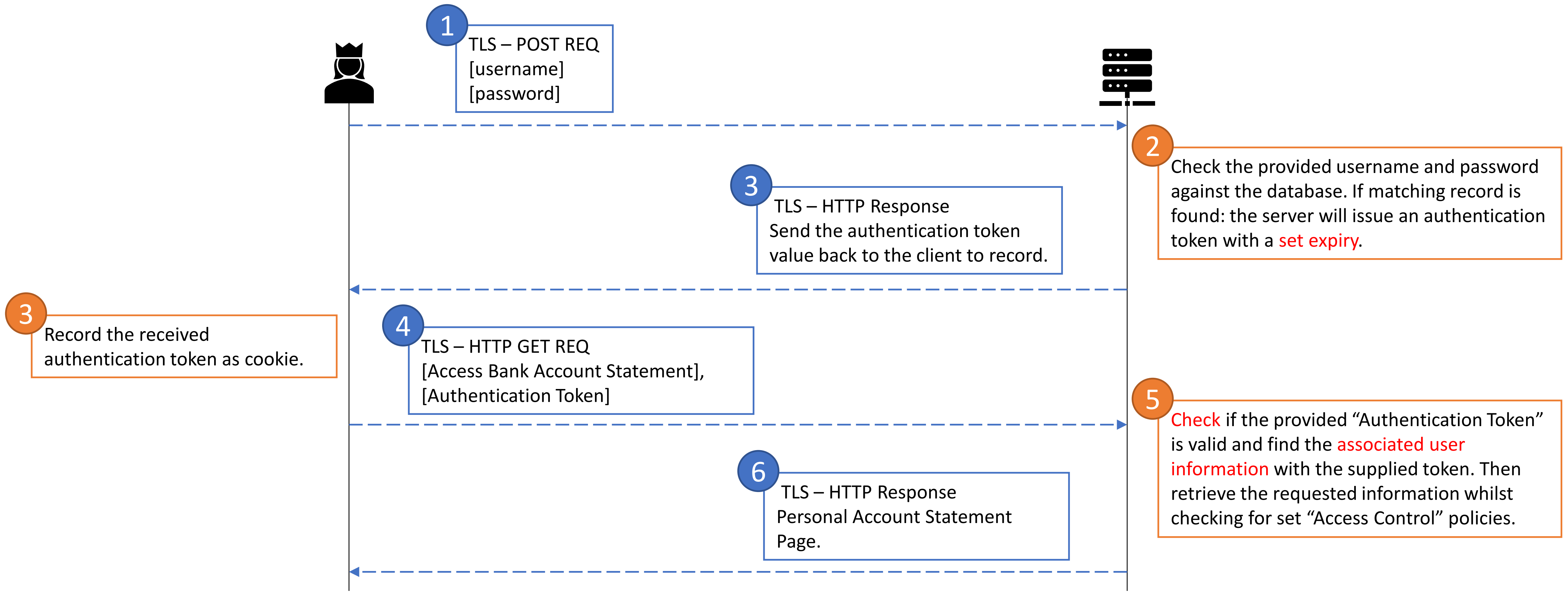
- Authentication Example – how can web server track if you have been logged in and authenticated for accessing your account's dashboard?
  - HTTP(s) protocol is stateless, which means, the web server has no way to identify who you are and what you have done in your previous interactions with the web server.
  - In order to keep track of your past actions and your identity, the web server can issue a "Session Token", link it to your account (i.e., your identity profile!), and previously actions that you have performed.
    - Think of Session Tokens as unique tracking number issued to you:
  - After receiving the session token from the web server, in every subsequent interaction with the web server, your browser will also send the session token with every HTTP request.

HTTP Cookies – cont’d

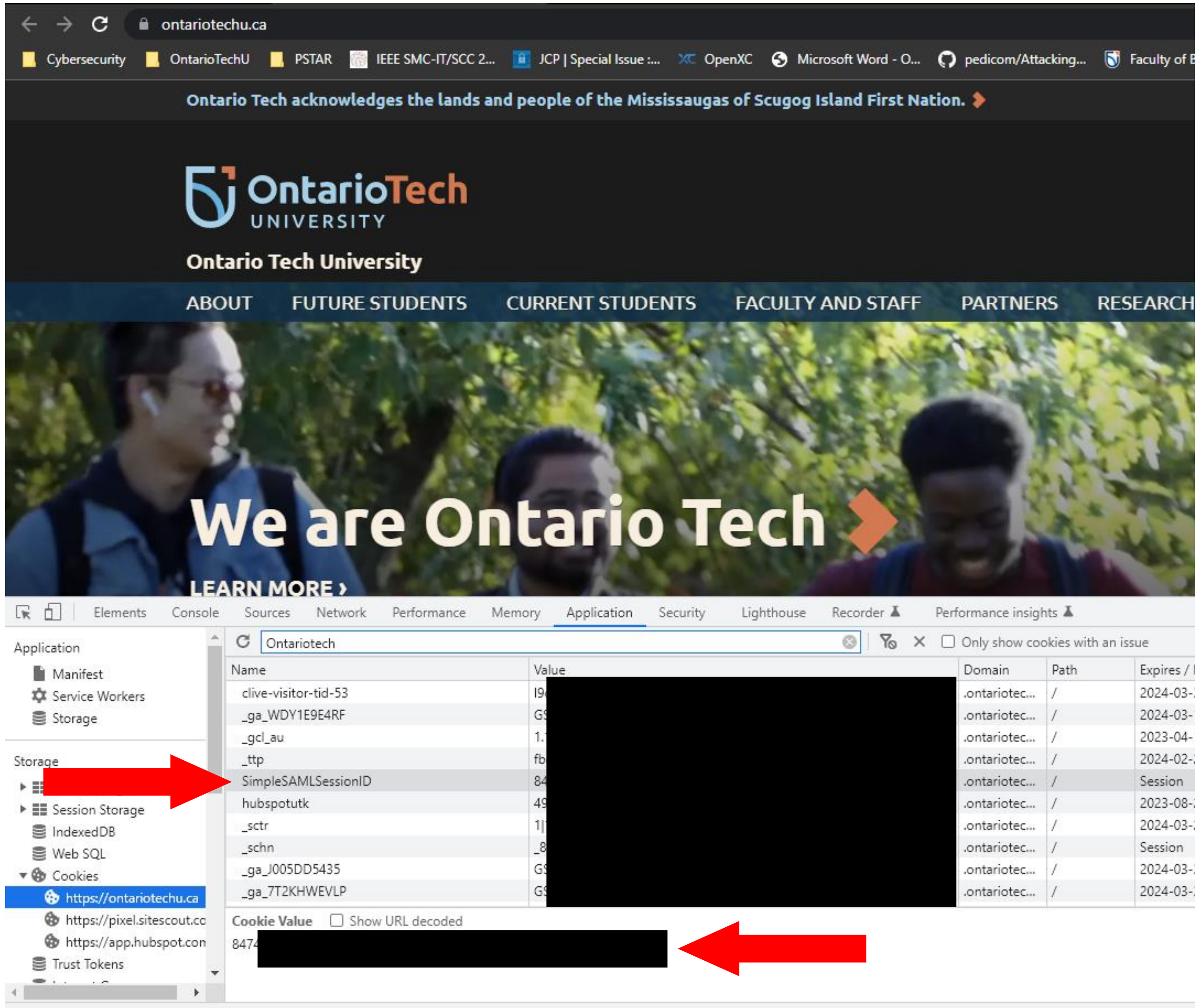
- Authentication Example – cont’d
  - The web server will authenticate the received session tokens in the HTTP requests:
    1. Is the received authentication token is associated with any user account in the database?
       - (if no, the token is invalid, the request is coming from a user that is not authenticated)
    2. Is the received authentication token considered expired?
       - (if yes, the token is invalid, the user must reauthenticate again!)
  - If all checks are passed, the session token will be used to perform access control checks and the weba app will handle the user request (e.g., accessing some confidential information!)

HTTP Cookies – cont'd

- Authentication Example – cont'd


1
TLS – POST REQ
[username]
[password]
2
Check the provided username and password against the database. If matching record is found: the server will issue an authentication token with a set expiry.
3
TLS – HTTP Response
Send the authentication token value back to the client to record.
3
Record the received authentication token as cookie.
4
TLS – HTTP GET REQ
[Access Bank Account Statement],
[Authentication Token]
5
Check if the provided "Authentication Token" is valid and find the associated user information with the supplied token. Then retrieve the requested information whilst checking for set "Access Control" policies.
6
TLS – HTTP Response
Personal Account Statement Page.

HTTP Cookies – cont’d



ontariotechu.ca

Cybersecurity | OntarioTechU | PSTAR | IEEE SMC-IT/SCC 2... | JCP | Special Issue :... | OpenXC | Microsoft Word - O... | pedicom/Attacking... | Faculty of B

Ontario Tech acknowledges the lands and people of the Mississaugas of Scugog Island First Nation.

OntarioTech UNIVERSITY

Ontario Tech University

ABOUT | FUTURE STUDENTS | CURRENT STUDENTS | FACULTY AND STAFF | PARTNERS | RESEARCH

We are Ontario Tech

LEARN MORE ›

Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse | Recorder | Performance insights

Application
- Manifest
- Service Workers
- Storage

Storage
- Session Storage
- IndexedDB
- Web SQL
- Cookies
  - https://ontariotechu.ca
  - https://pixel.sitescout.co
  - https://app.hubspot.con
- Trust Tokens

Ontariotech | Only show cookies with an issue

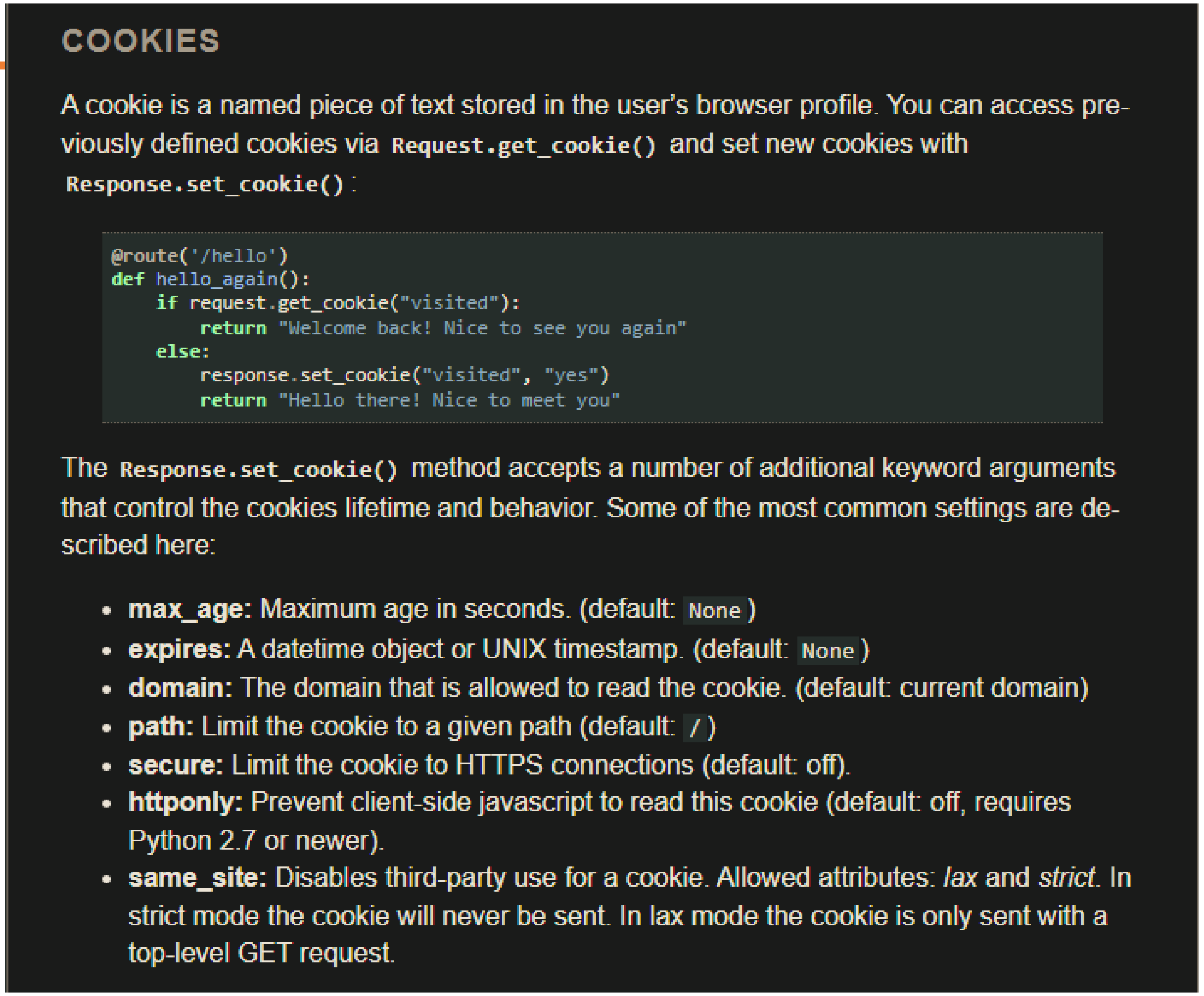
Name	Value	Domain	Path	Expires / I
clive-visitor-tid-53	I9[redacted]	.ontariotec...	/	2024-03-
_ga_WDY1E9E4RF	G[redacted]	.ontariotec...	/	2024-03-
_gcl_au	1.[redacted]	.ontariotec...	/	2023-04-
_ttp	fb[redacted]	.ontariotec...	/	2024-02-
SimpleSAMLSessionID	84[redacted]	.ontariotec...	/	Session
hubspotutk	49[redacted]	.ontariotec...	/	2023-08-
_sctr	1[redacted]	.ontariotec...	/	2024-03-
_schn	_8[redacted]	.ontariotec...	/	Session
_ga_J005DD5435	G[redacted]	.ontariotec...	/	2024-03-
_ga_7T2KHWEVLP	G[redacted]	.ontariotec...	/	2024-03-

Cookie Value | Show URL decoded

8474[redacted]

HTTP Cookies

- Security Tips:
  - Bear in mind the cookies are stored on clients' devices and may be accessible by other applications and individuals.
  - Critical/confidential cookie values must be encrypted and not in plane text.
  - Session tokens (e.g., authentication tokens) must have expiry date and set as HTTP only so that no JavaScript code could access them (protecting against CSRF attacks!)
  - Do not assume the received cookie values are correct (on the server side) and make sure to perform necessary checks against received cookies!



COOKIES

A cookie is a named piece of text stored in the user's browser profile. You can access previously defined cookies via `Request.get_cookie()` and set new cookies with `Response.set_cookie()`:

```
@route('/hello')
def hello_again():
    if request.get_cookie("visited"):
        return "Welcome back! Nice to see you again"
    else:
        response.set_cookie("visited", "yes")
        return "Hello there! Nice to meet you"
```

The `Response.set_cookie()` method accepts a number of additional keyword arguments that control the cookies lifetime and behavior. Some of the most common settings are described here:

- **max_age:** Maximum age in seconds. (default: `None`)
- **expires:** A datetime object or UNIX timestamp. (default: `None`)
- **domain:** The domain that is allowed to read the cookie. (default: current domain)
- **path:** Limit the cookie to a given path (default: `/`)
- **secure:** Limit the cookie to HTTPS connections (default: off).
- **httponly:** Prevent client-side javascript to read this cookie (default: off, requires Python 2.7 or newer).
- **same_site:** Disables third-party use for a cookie. Allowed attributes: *lax* and *strict*. In strict mode the cookie will never be sent. In lax mode the cookie is only sent with a top-level GET request.

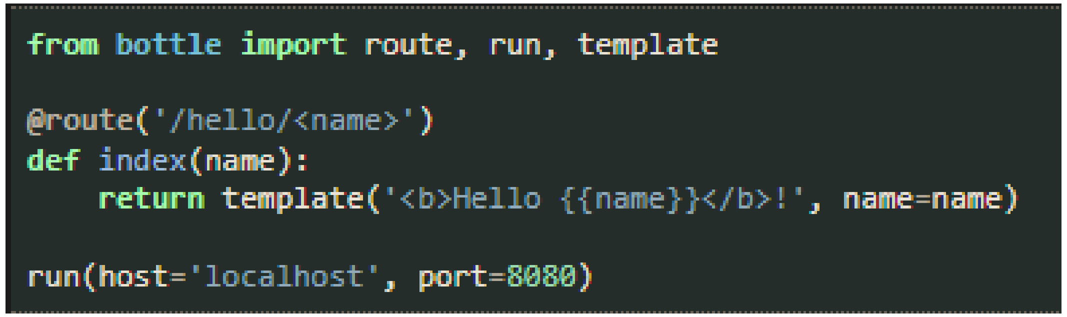
https://bottlepy.org/docs/dev/tutorial.html

OntarioTech Business & IT

Setting Up

- In this module we are going to setup a web development environment using
  - Python 3
    - Comes with Kali Linux
  - Bottle: Python Web Framework
    - https://bottlepy.org/docs/dev/
    - Install using `pip`: `pip install bottle`
  - Kali Linux (32 or 64 bit, doesn't really matter)

- Your first web app
  - Create hello_web.py with the following code:



```python
from bottle import route, run, template

@route('/hello/<name>')
def index(name):
    return template('<b>Hello {{name}}</b>!', name=name)

run(host='localhost', port=8080)
```

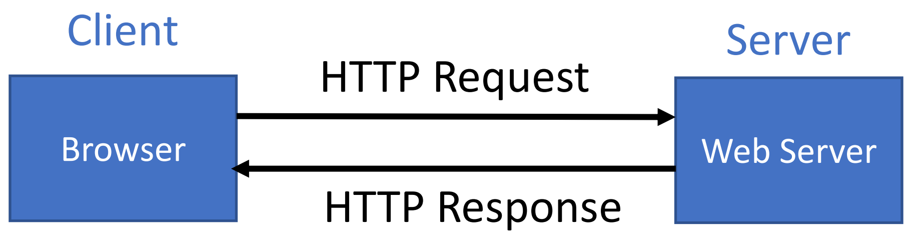
  - Run your web app using:

    `python hello_web.py`

  - Navigate to your running web app on a local browser of your Kali:

    http://localhost:8080/YourNameHere

Setting Up – cont'd

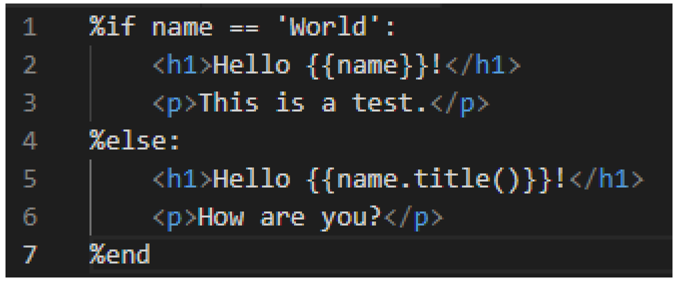


- Important Notes:
  - Although in most real-world applications clients and servers are physically separate, in this configuration, we are running them both on your Kali VM. They are considered (logically) separate entities.
  - The code (python) that runs on the web server is called "server-side or back-end code."
  - The code (e.g., JavaScript) that runs on the browser is called "client-side or front-end code."
  - Bottle comes with a development web server. When you run your web application (as we did in the previous slide) the web server is launched and your python code automatically is hosted on the running web server.
  - http://localhost:8080/...
    - "localhost" is a self referencing host name. It points to your own machine/environment.
    - We are running your web server on the local port 8080 instead of conventional port 80. That's why we have to specify it in the URL.

OntarioTech Business & IT

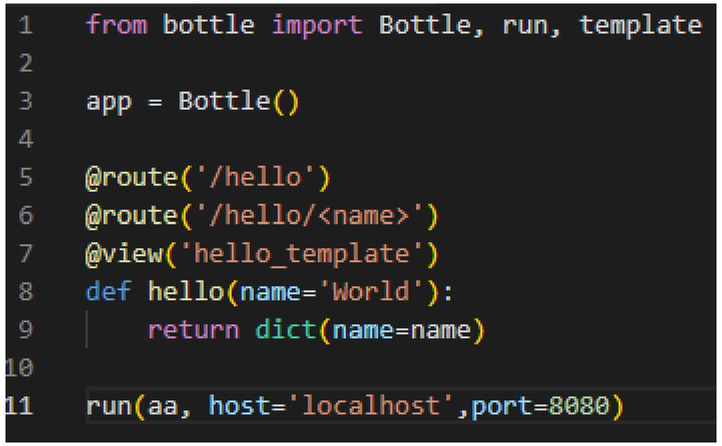
Setting Up – cont'd

- You can separate (and almost always practiced) your Python code from HTML template of your web app.
  - For better maintainability, readability of your Web App.
  - Template files must be stored with ".tpl" extension
    - Example: hello_template.tpl



```
%if name == 'World':
    <h1>Hello {{name}}!</h1>
    <p>This is a test.</p>
%else:
    <h1>Hello {{name.title()}}!</h1>
    <p>How are you?</p>
%end
```

- Then, you can reference the template file in your code and pass in the required parameters.

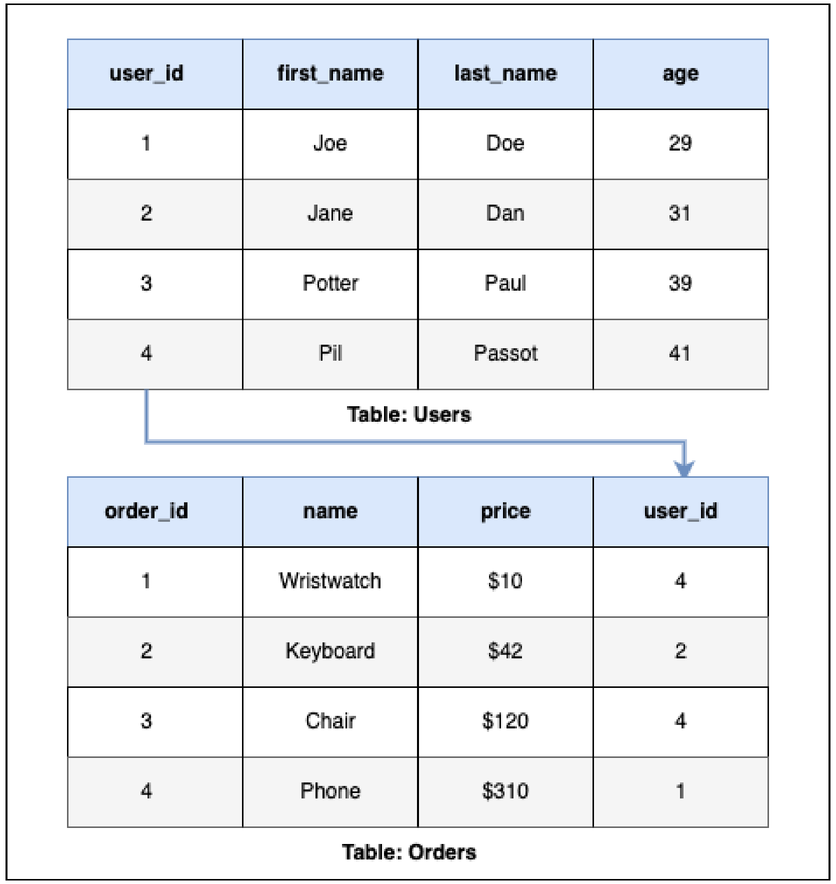


```
from bottle import Bottle, run, template

app = Bottle()

@route('/hello')
@route('/hello/<name>')
@view('hello_template')
def hello(name='World'):
    return dict(name=name)

run(aa, host='localhost',port=8080)
```

OntarioTech
Business & IT

SQL Injection

- Databases are one of the primary places for storing large quantities of (user) data.
  - Structured Query Language (SQL) statements/queries are often used to access and/or manipulate the data stored in some notable relational database management systems (DBMS). E.g., MS-SQL Server, MySQL, Oracle DB, etc.
  - Almost all business applications (not just specific to Web Apps) utilize DBMS to store their user/business data.
- What's a relational database
  - Is used to store data items with pre-defined relationship between them.
  - Items are organized as a set of tables with columns and rows.



user_id	first_name	last_name	age
1	Joe	Doe	29
2	Jane	Dan	31
3	Potter	Paul	39
4	Pil	Passot	41

Table: Users

order_id	name	price	user_id
1	Wristwatch	$10	4
2	Keyboard	$42	2
3	Chair	$120	4
4	Phone	$310	1

Table: Orders

https://www.freecodecamp.org/news/what-is-sql-database-definition-for-beginners/

OntarioTech Business & IT

SQL Injection – cont'd

- SQL Queries
  - Inserting into a table:

```
INSERT INTO Users (‘first_name’,’last_name’,’age’)
VALUES(‘Joe’,’Doe’,’29’);
```

  - Select some data from a table:

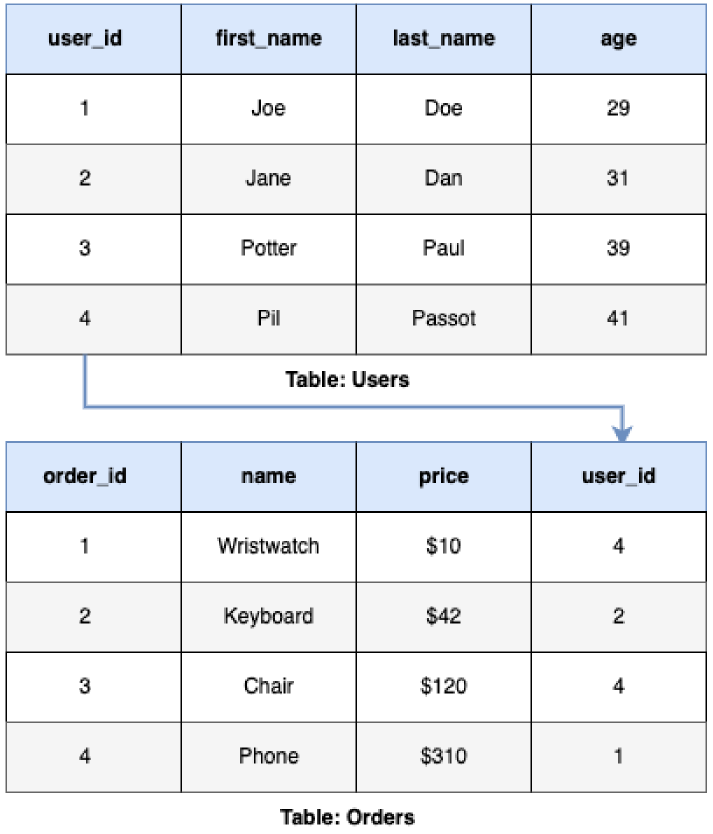
```
SELECT first_name, last_name  FROM Users
WHERE age >= 39;
```

  - Delete some data from a table:

```
DELETE FROM Orders WHERE order_id=2;
-- deleting some customer order!
```

    - Double dash-lines are for writing comments in a SQL query.

https://www.sqlitetutorial.net/ for more information!



user_id	first_name	last_name	age
1	Joe	Doe	29
2	Jane	Dan	31
3	Potter	Paul	39
4	Pil	Passot	41

Table: Users

order_id	name	price	user_id
1	Wristwatch	$10	4
2	Keyboard	$42	2
3	Chair	$120	4
4	Phone	$310	1

Table: Orders

https://www.freecodecamp.org/news/what-is-sql-database-definition-for-beginners/

OntarioTech Business & IT

SQL Injection – cont’d

- The SQL injection attack is
  - Performed by a threat actor inputting a specific string to manipulate the SQL statement to do something other than that intended by the programmer!
  - Example: suppose we are going to remove a specific customer order from our database by reading the “order_id” as an input parameter from user and stored in a python variable called `del_order_id`

```
sql_query = f”DELETE FROM Orders WHERE order_id={del_order_id}”
```

    - Suppose the threat actor provides the following order id to be deleted: 1 or 1=1
    - The SQL query constructed using the given input will look like:

```
sql_query = f”DELETE FROM Orders WHERE order_id=1 or 1=1”
```

    - This will result in deletion of every record in “Orders” table! [Why? In class discussion]

SQL Injection – cont’d

- “The attack methodology
  1. Test input values to see if manipulation of SQL is possible!
  2. Experiment with SQL inputs, using error messages to enumerate the database and provide exploitation information.
     - *[Even if the error messages are suppressed, the attacker can still make some guesses]*
     - *[The guess work becomes a lot easier when the codebase is Open Source!]*
  3. Craft a SQL exploit input to achieve the exploitation goal” – CSSLP by Conklin and Shoemaker, Chapter 9

- Affects data integrity and availability. [why? In class discussion]
  - and is a server-side attack when discussed in the context of web applications.

SQL Injection – cont’d

- Mitigation and Defense Techniques:
  - Design SQL queries in a manner that does not allow the query string to be manipulated by the user. [not always possible!]
  - Use stored procedures for all accesses and manipulations, with user input being in the form of variables passed to the stored procedure.
  - Use “parametrized queries” – totally opposite of string concatenation!
    - `db.execute_sql('SELECT * FROM series(?, ?, ?)', (0, 5, 2))`
  - Object Relational Mapping (ORM) libraries to sanitize and generate SQL queries for you (e.g., peebee – we will be using this in our Assignment #2)
- The primary mitigation for this attack is developer/programmer awareness!

OntarioTech Business & IT

Cross-Site Scripting (XSS)

- Sometimes user inputs must be displayed as part of a web page.
  - Think of comments posted by some users under your YouTube videos!
  - The input “texts” (e.g., posted) will be rendered and displayed by your web browser.
  - Thus, any HTML/Script code in the posted texts by malicious users, may be also executed on your browser!
- What can a threat actor accomplish by executing a JavaScript/HTML code on your browser?
  - Theft of authentication information
  - Session hijacking
  - Deploy hostile content
  - Change user settings
  - Impersonate a use

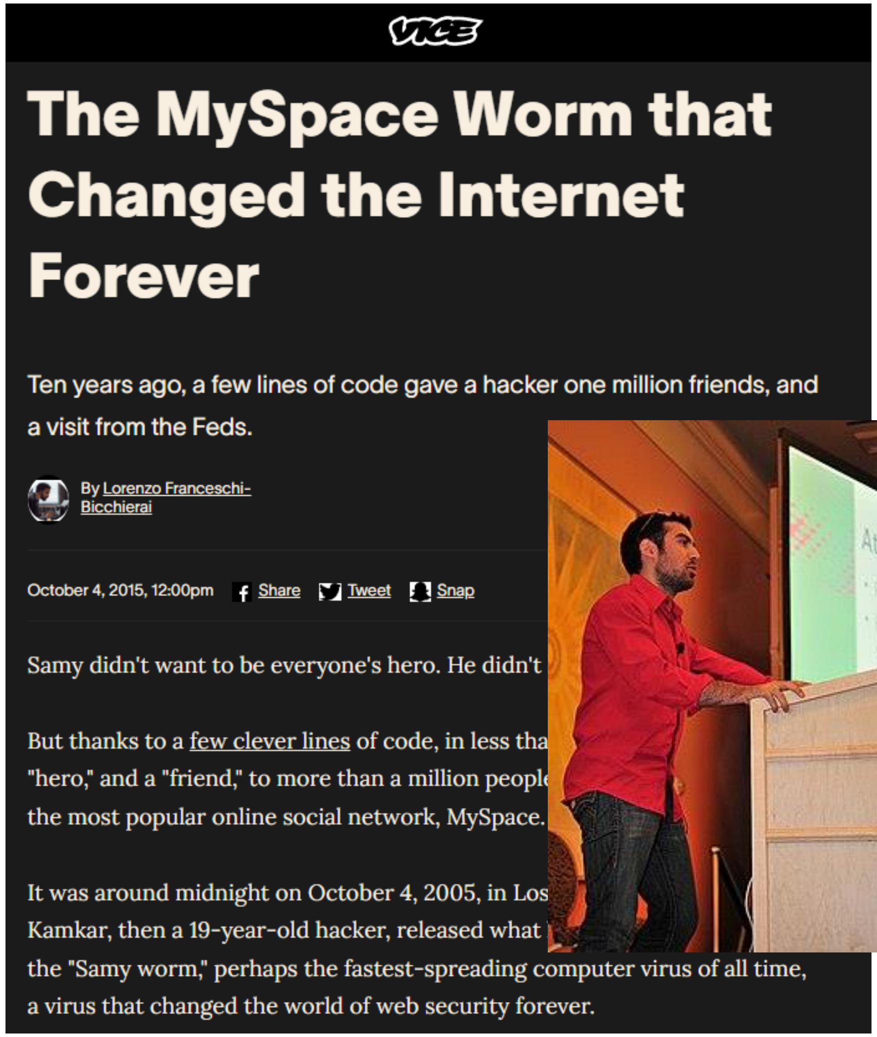
Cross-Site Scripting (XSS) - cont’d

- Stored XSS
  - The malicious client-side script is stored somewhere in the vulnerable web application (or its dependencies).
  - The threat-actor managed to provide an input to the web app that is stored and later-on displayed on the browser of regular users. Must contain some scripting elements!
  - https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/02-Testing_for_Stored_Cross_Site_Scripting
- Reflected XSS
  - Is not stored in the application, but the script is provided to the web application via the URL.
  - E.g., www.abc.com/say_hello.php?name=PooriaMadani – that says hello the name passed in the GET parameter. (The passed name will become part of the HTML document.)
  - V.s. www.abc.com/say_hello.php?name=<script>alert(“Pawned!”);</script>
  - Passed to the victim via social engineering – getting the victim to click on a crafted URL by the threat actor!
  - https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_Reflected_Cross_Site_Scripting

Cross-Site Scripting (XSS) – cont'd

- Case Study – Sammy Worm!
  - Within 20 hours (back in October 2005) got executed and changed profile settings of one million MySpace users.
  - Sentenced to no Internet use for three years, 90 days' community service and $20,000 in restitution!
  - The story told by the hacker himself!
    - https://samy.pl/myspace
  - Some of the technical details how he did it
    - https://samy.pl/myspace/tech.html
  - An awesome Interview with him

    https://www.youtube.com/watch?v=DtnuaHl378M



VICE

The MySpace Worm that Changed the Internet Forever

Ten years ago, a few lines of code gave a hacker one million friends, and a visit from the Feds.

By Lorenzo Franceschi-Bicchierai

October 4, 2015, 12:00pm Share Tweet Snap

Samy didn't want to be everyone's hero. He didn't

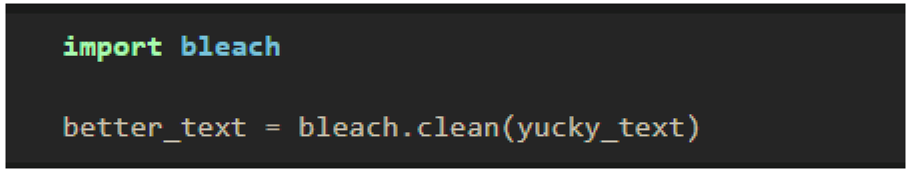
But thanks to a few clever lines of code, in less tha
"hero," and a "friend," to more than a million peopl
the most popular online social network, MySpace.

It was around midnight on October 4, 2005, in Los
Kamkar, then a 19-year-old hacker, released what
the "Samy worm," perhaps the fastest-spreading computer virus of all time, a virus that changed the world of web security forever.

Cross-Site Scripting (XSS) – cont'd

- Mitigation and Defense Techniques:
  - Do not trust user inputs, make sure it is empty of any HTML and/or JavaScript tags.

    Specially if they are going to be displayed in an HTML page later on!
  - Often, it is taken care of by modern Web App Frameworks such Bottle.py – do not assume, however, always confirm!
  - Use third party input cleansing libraries:

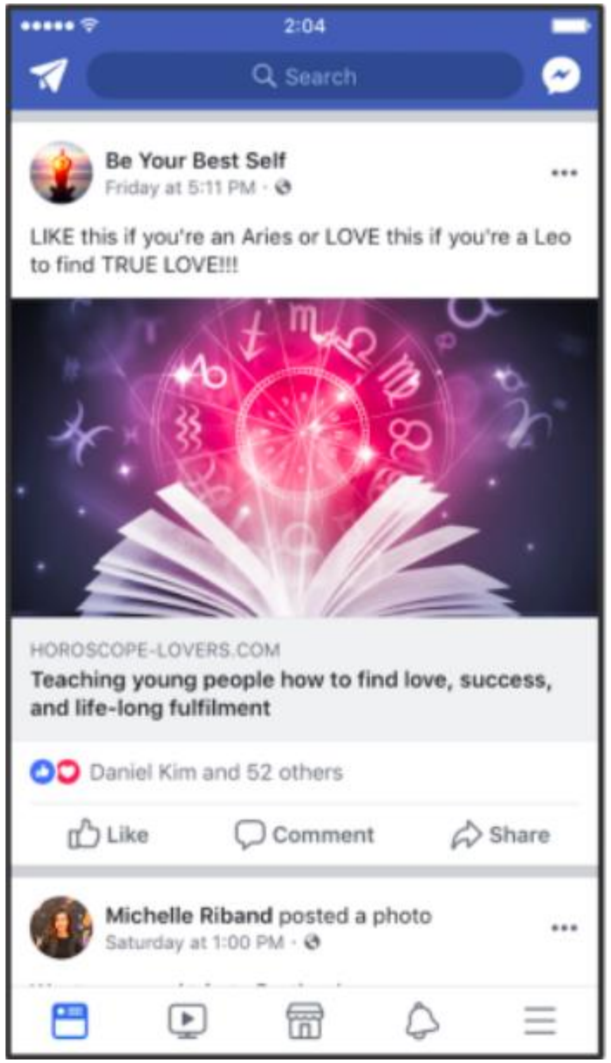


```
import bleach

better_text = bleach.clean(yucky_text)
```

- Affect data confidentiality, integrity and availability. [why? In class discussion]
  - and is a client-side attack

Cross-Site Request Forgery (CSRF)

- Is an attack that forces victims to execute unintended actions on a web application in which they are currently authenticated.


Be Your Best Self
Friday at 5:11 PM ·

LIKE this if you're an Aries or LOVE this if you're a Leo to find TRUE LOVE!!!

HOROSCOPE-LOVERS.COM
Teaching young people how to find love, success, and life-long fulfilment

Daniel Kim and 52 others

Michelle Riband posted a photo
Saturday at 1:00 PM ·

OntarioTech Business & IT

Cross-Site Request Forgery (CSRF) – cont’d

- Suppose that your bank has designed a web service for its clients to make online e-transfers by forming and sending the following HTTP Get request:
  - http://bank.com/transfer.php?amount=999&to=intended@email.address
  - Works only if you have previously logged in to Bank.com and your login session is still active.

- CSRF is exploiting behaviour that are considered proper and well defined
  - Under circumstances outside the authorized use.
  - For example, when attacker get you to visit the following link (given you are already logged in to your banking website in another browser tab)
    - http://bank.com/transfer.php?amount=999&to=attacker@email.address

Cross-Site Request Forgery (CSRF) – cont’d

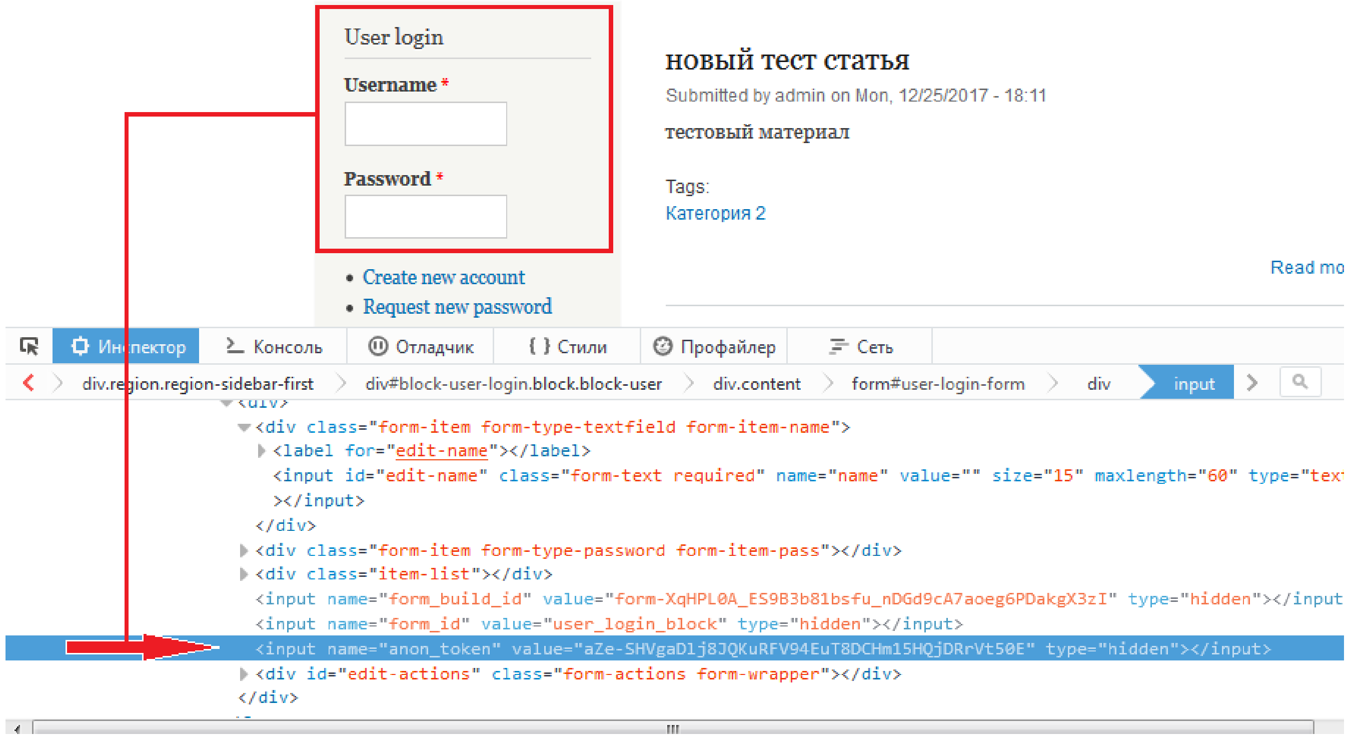
- How threat actors can trick users to make such HTTP requests?
  - Social engineering: embed request links into comments, email, and documents.
  - XSS: create CSRF HTTP requests on behalf o the victim by necessary embedding JavaScript codes into one of the web pages that the victim visits.

- Affect data confidentiality, integrity and availability. [why? In class discussion]
  - and is a client-side attack

Cross-Site Request Forgery (CSRF) – cont’d

Mitigation and Defense Techniques:

- Make sure important web requests that have side-effects (i.e., changing the state of stored information) are only accepted as “POST requests”.
- Use random token values that must be submitted along with each POST. requests (known as CSRF token).
    - The token is used to make sure that submitted POST request is generated from a recently retrieved HTML page from the server and was not generated on the fly by a malicious JavaScript code!



новый тест статья

Submitted by admin on Mon, 12/25/2017 - 18:11

тестовый материал

Tags:
Категория 2

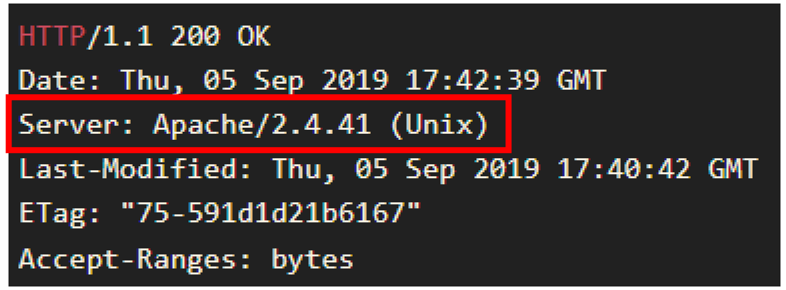
div.region.region-sidebar-first > div#block-user-login.block.block-user > div.content > form#user-login-form > div > input

```
    <div>
      <div class="form-item form-type-textfield form-item-name">
        <label for="edit-name"></label>
        <input id="edit-name" class="form-text required" name="name" value="" size="15" maxlength="60" type="tex
        ></input>
      </div>
      <div class="form-item form-type-password form-item-pass"></div>
      <div class="item-list"></div>
      <input name="form_build_id" value="form-XqHPL0A_ES9B3b81bsfu_nDGd9cA7aoeg6PDakgX3zI" type="hidden"></input
      <input name="form_id" value="user_login_block" type="hidden"></input>
      <input name="anon_token" value="aZe-SHVgaD1j8JQKuRFV94EuT8DCHm15HQjDRrVt50E" type="hidden"></input>
      <div id="edit-actions" class="form-actions form-wrapper"></div>
    </div>
```

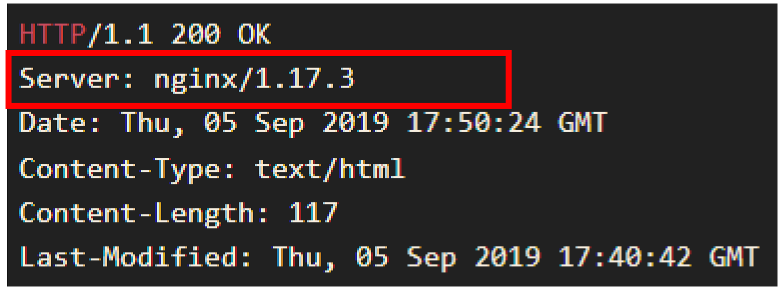
https://www.drupal.org/project/anonymous_token

Web Server Fingerprinting

- Is the task of identifying the type and version of the hosting web server.
  - Why?
    - Can enable threat actors to determine if the hosting web server is vulnerable to a specific attack.
- Using HTTP Response Headers
  - Examples:



```
HTTP/1.1 200 OK
Date: Thu, 05 Sep 2019 17:42:39 GMT
Server: Apache/2.4.41 (Unix)
Last-Modified: Thu, 05 Sep 2019 17:40:42 GMT
ETag: "75-591d1d21b6167"
Accept-Ranges: bytes
```

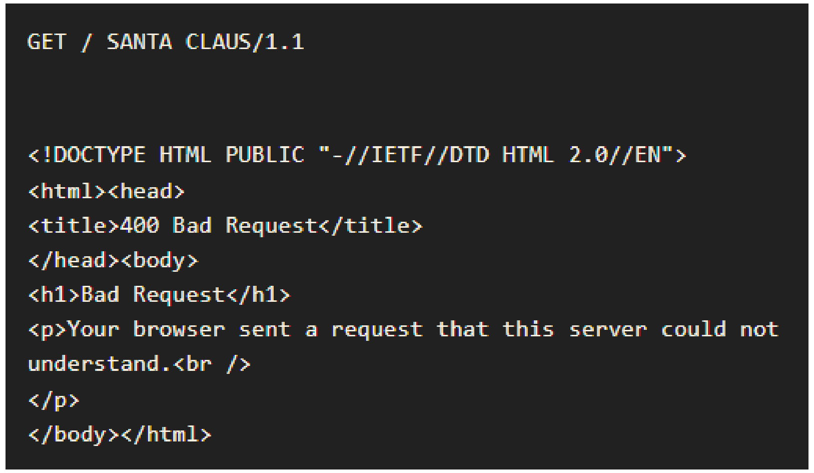


```
HTTP/1.1 200 OK
Server: nginx/1.17.3
Date: Thu, 05 Sep 2019 17:50:24 GMT
Content-Type: text/html
Content-Length: 117
Last-Modified: Thu, 05 Sep 2019 17:40:42 GMT
```

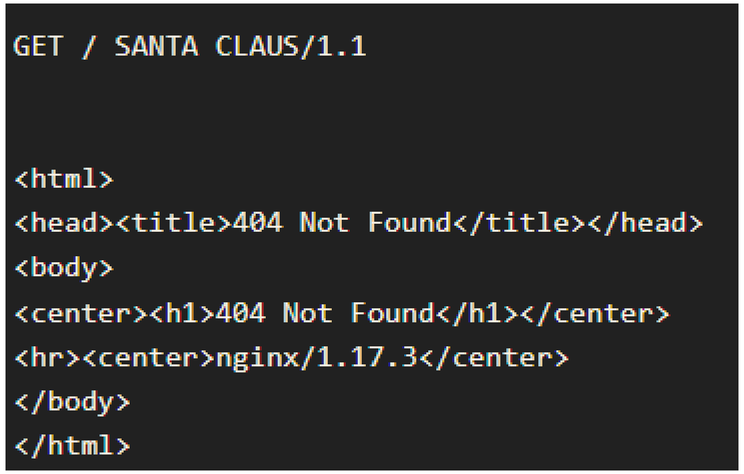
    - Make note of "Server" attribute and also, the order of the attributes (this code be telling!)
    - Sure, it is advised to change the default Server attribute to something else!

Web Server Fingerprinting – cont'd

- Sending Malformed Requests
  - Some servers may be identified by examining their "default" error messages in response to malformed request. E.g., requesting a page that does not exist!



```
GET / SANTA CLAUS/1.1


<!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>
</body></html>
```



```
GET / SANTA CLAUS/1.1


<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.17.3</center>
</body>
</html>
```

https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server

OntarioTech Business & IT

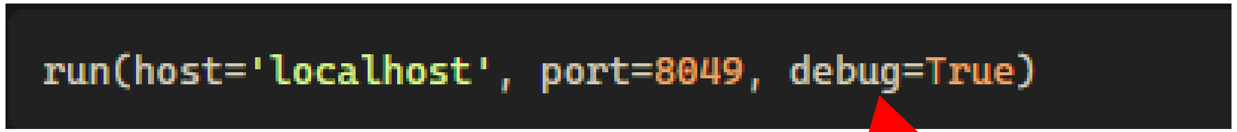
Web Server Fingerprinting – cont'd

- Remediation
  - Exposed server information is not necessarily a vulnerability in itself!
  - However, it can assist attackers in exploiting other vulnerabilities that may exist!

  - You may change default attribute values and error messages.
  - Use reverse proxy servers to create an additional layer of security between the web server and the Internet.
  - Use forged HTTP response attributes (e.g., Server: "nginx" when in reality you are running an Apache!)
  - https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server

Improper Error Handling

- Error messages can sometimes give away a lot of important information (E.g., database scheme) of your web application.
- Disable detailed error messages to be exposed the end-user when deploying your web application.



```
run(host='localhost', port=8049, debug=True)
```

Ok during development but not for deployment!

OWASP

- “The Open Worldwide Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.” -- https://owasp.org/
- They have an extensive “Web Application Security Testing” framework that explain most common web app vulneraries, how test for them, and remediation strategies.
  - It will be part of assignment #2.
  - https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/