Web Security Project

Georgia Tech

Project Overview

- Attacks on 3 common web vulnerabilities and their defenses:
  - SQL Injection
  - XSS
  - CSRF
- Familiarity with Web Inspector, HTML, JavaScript and SQL programming
- Clone the starter code repo to set up the Docker container!
- Vulnerable website “BuzzBuzzGo” with Firefox 91 (running inside Docker)
- Submission on https://autograder.gtinfosec.org/

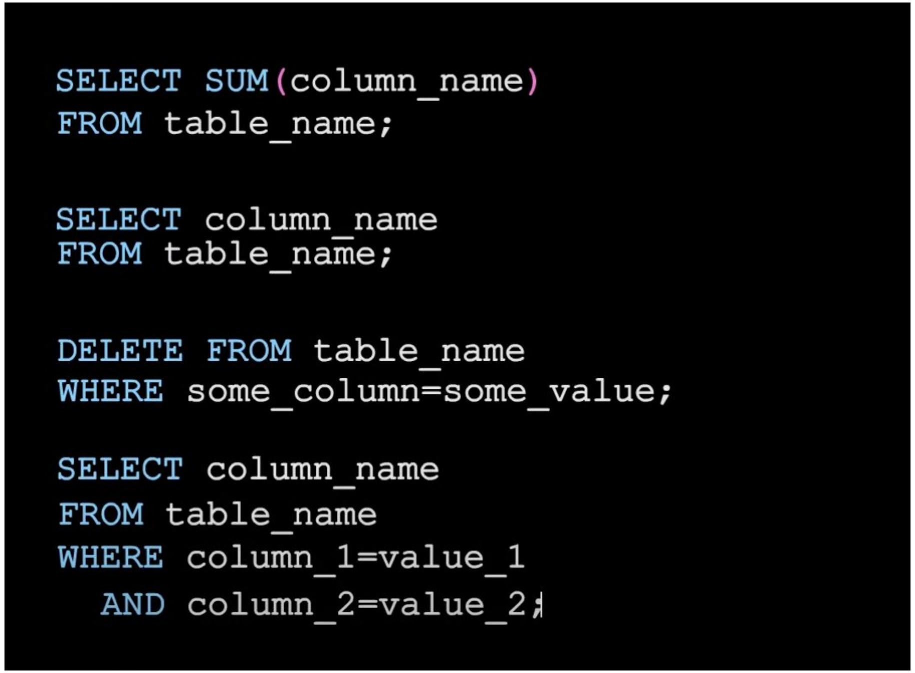
Project Overview

WARNING!!!

DO NOT ATTEMPT THESE ATTACKS ON WEBSITES WITHOUT EXPLICIT AUTHORIZATION

SQL Commands

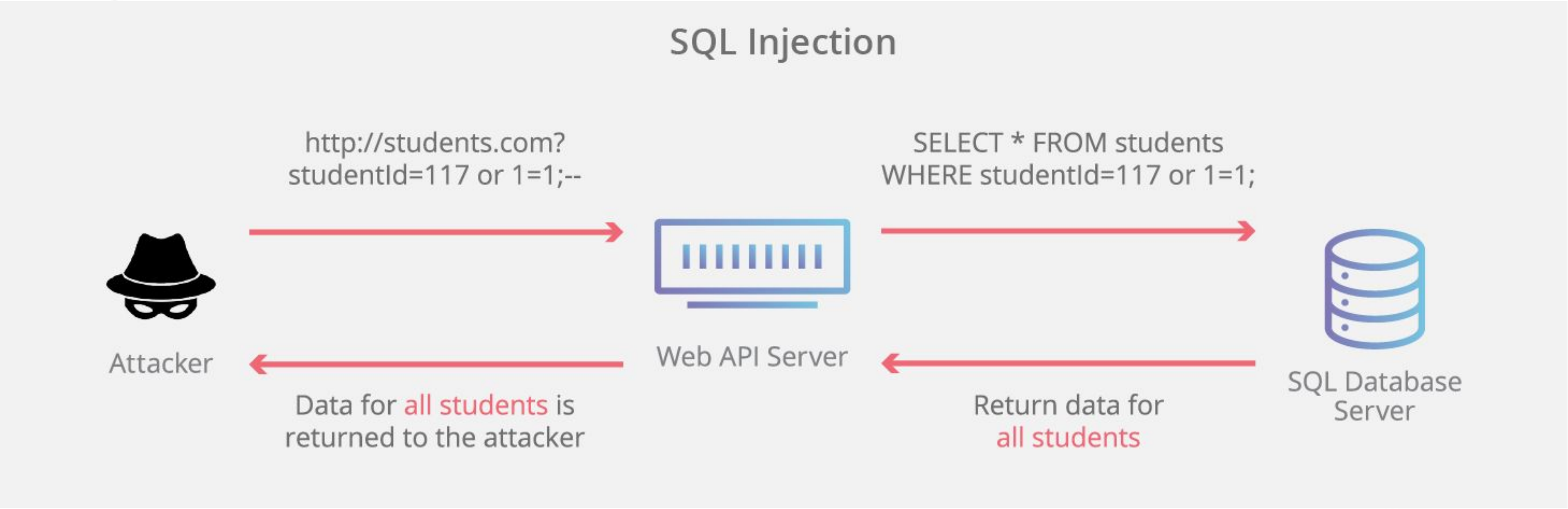
Normal SQL statements can be used to performs actions on the database like storing, retrieving, and deleting data



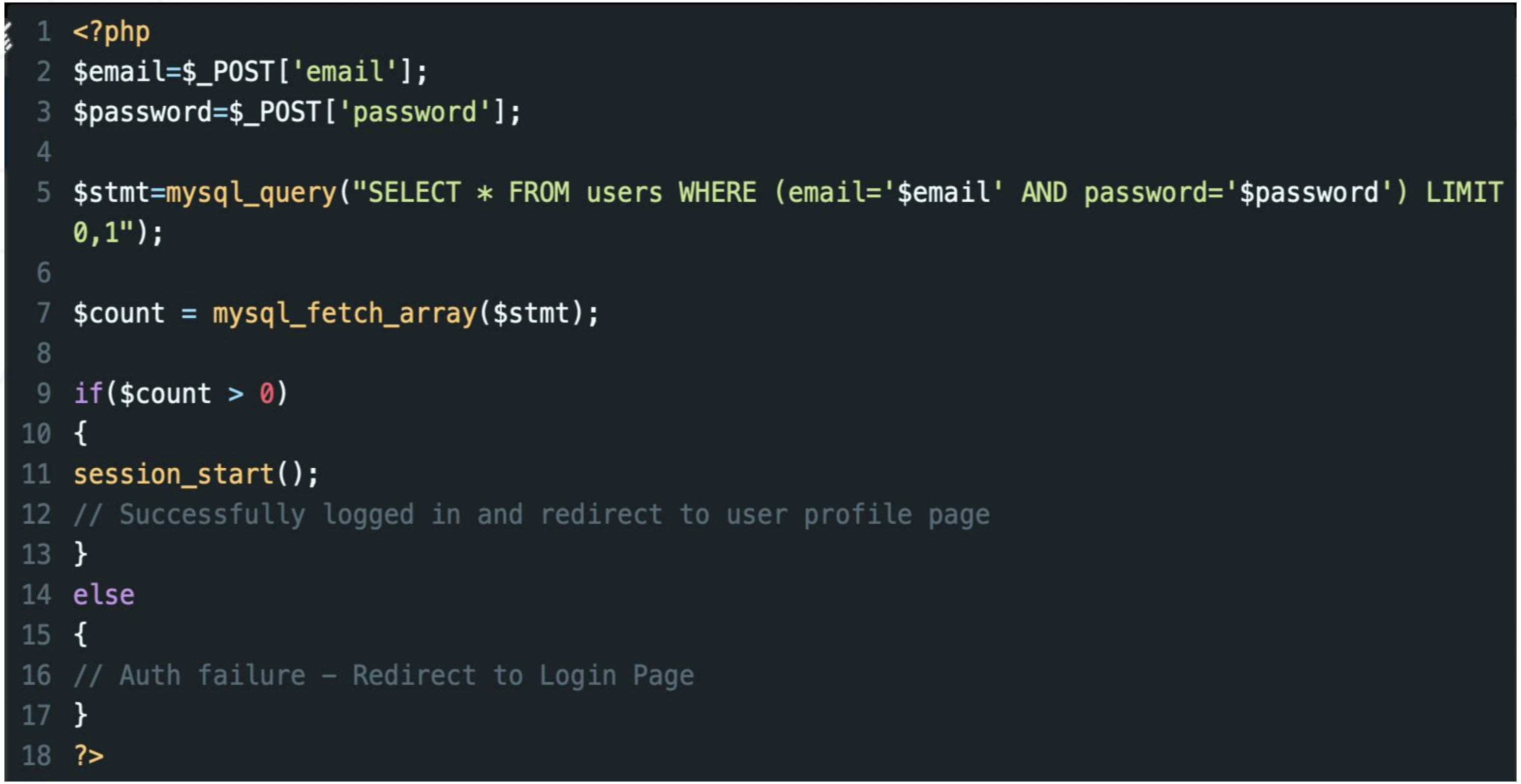
```
SELECT SUM(column_name)
FROM table_name;

SELECT column_name
FROM table_name;

DELETE FROM table_name
WHERE some_column=some_value;

SELECT column_name
FROM table_name
WHERE column_1=value_1
  AND column_2=value_2;
```

SQL Injection

- A common attack vector that uses malicious SQL code for backend database manipulation intending to access unauthorized information.


SQL Injection
http://students.com?studentId=117 or 1=1;--
SELECT * FROM students WHERE studentId=117 or 1=1;
Attacker
Web API Server
SQL Database Server
Data for all students is returned to the attacker
Return data for all students

SQL Injection Walkthrough - Vulnerable code



```php
<?php
$email=$_POST['email'];
$password=$_POST['password'];

$stmt=mysql_query("SELECT * FROM users WHERE (email='$email' AND password='$password') LIMIT 0,1");

$count = mysql_fetch_array($stmt);

if($count > 0)
{
session_start();
// Successfully logged in and redirect to user profile page
}
else
{
// Auth failure - Redirect to Login Page
}
?>
```

SQL Injection Tasks

- Goal – Use SQL Injection to successfully login as user ‘victim’
- Task 1.0 – No Defenses:
  - The password field is simply enclosed in single quotes
- Task 1.1 – Simple Escaping:
  - Server escapes single quotes by replacing with double quotes
  - Hint: Think of other ways to bypass this naïve sanitization approach
- Task 1.2 – Escaping and Hashing:
  - Server escaped username and uses MD5 to hash the password
  - Hint: Think of how cryptographically unsafe MD5 is! Make a program that generates hashes and searches for an injection

Cross-Site Scripting Attacks (XSS)

- Websites that accept unsanitized and unvalidated user input are setting themselves up for disaster.
  - Web browsers can misinterpret user input as code!

- Cybersecurity for Web Developers 101: **Do not expect users to provide appropriate input when interacting with a website!**
  - Best Case: Benign users *accidentally* break the site

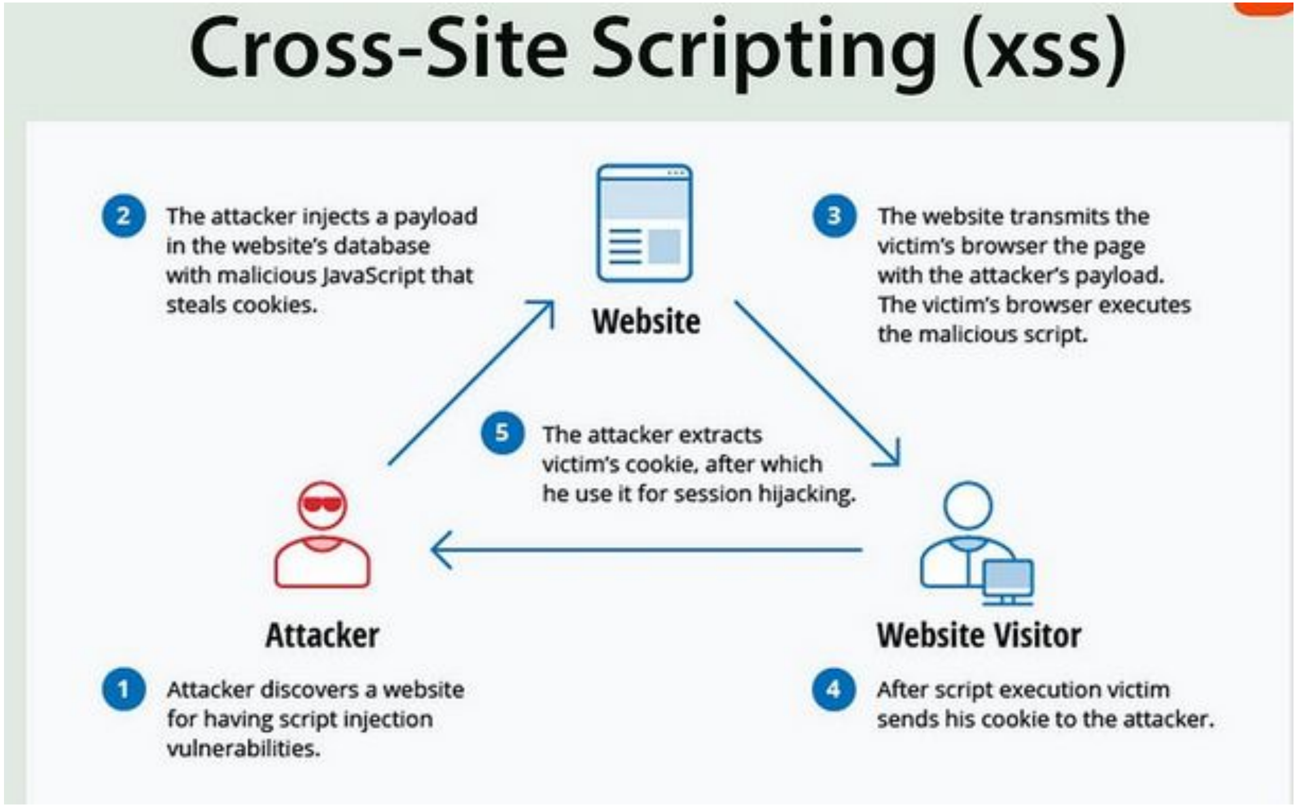
  - Worst Case: Attackers intentionally issue input to wreak havoc

Cross-Site Scripting Attacks (XSS)

- **Benign users** visit websites believing the web page being loaded by their browser is safe.
  - Website may serve user-generated data which, without proper sanitation, can be misinterpreted and executed
  - Browsers execute scripts on web page without verifying the scripts are safe

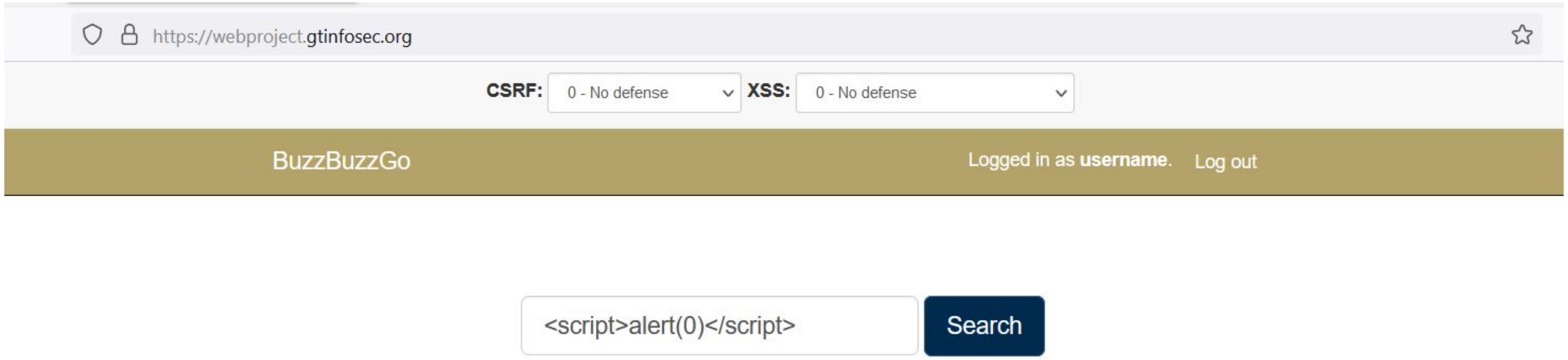
- **Attackers** inject into a legitimate web page a malicious script that gets loaded with the rest of the page. Popular ways of script injection are the following:
  - Issuing web requests containing malicious code written in JavaScript and HTML
  - Adding malicious code to the end of URLs

General Idea of XSS Attacks


Cross-Site Scripting (xss)
Website
Attacker
Website Visitor
1 Attacker discovers a website for having script injection vulnerabilities.
2 The attacker injects a payload in the website's database with malicious JavaScript that steals cookies.
3 The website transmits the victim's browser the page with the attacker's payload. The victim's browser executes the malicious script.
4 After script execution victim sends his cookie to the attacker.
5 The attacker extracts victim's cookie, after which he use it for session hijacking.

What do we mean by script?

- A body of Javascript code between two script tags “<script> ….. </script>” that gets executed by the browser when input into a textbox


https://webproject.gtinfosec.org

CSRF: 0 - No defense XSS: 0 - No defense

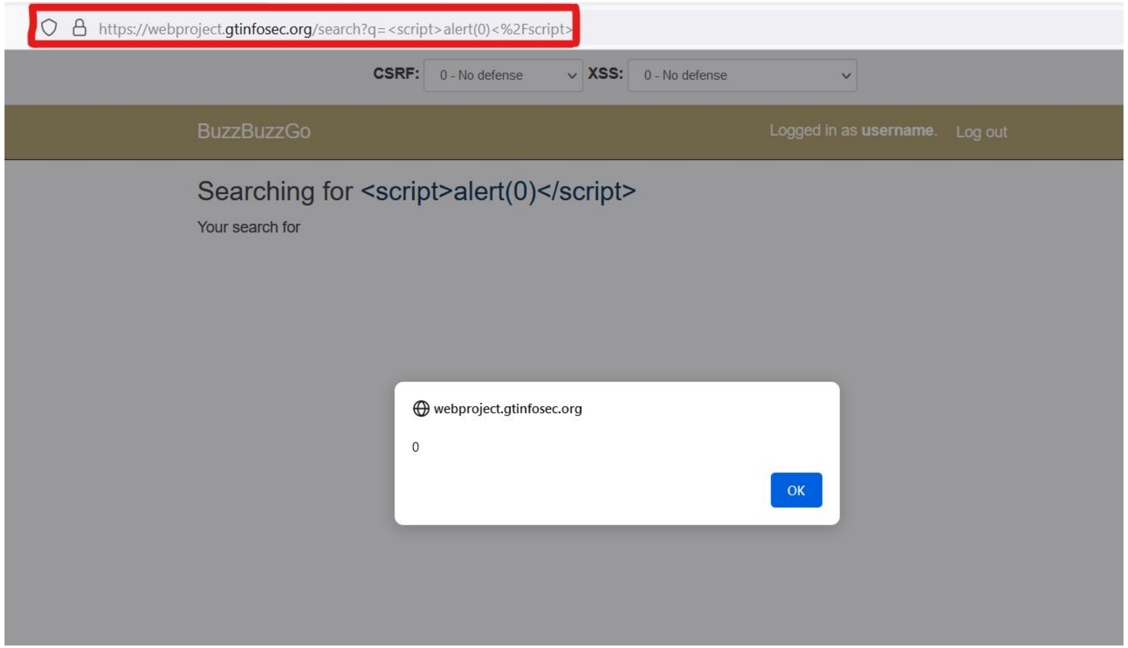
BuzzBuzzGo Logged in as **username**. Log out

```
<script>alert(0)</script>
```

Simple “alert” script typed into textbox on the project website

What happens when the script is run?

When the script gets executed by clicking the Search button, notice the activity that happens. Where did this dialog box come from? How did URL change?



https://webproject.gtinfosec.org/search?q=<script>alert(0)<%2Fscript>

CSRF: 0 - No defense XSS: 0 - No defense

BuzzBuzzGo Logged in as **username**. Log out

Searching for <script>alert(0)</script>

Your search for

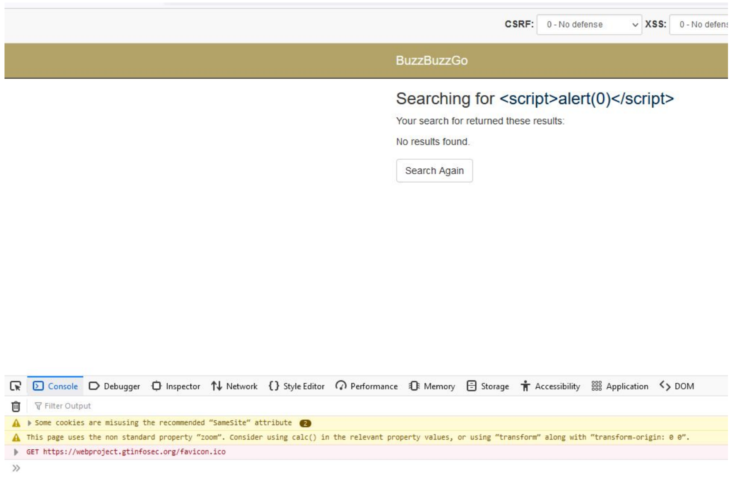
webproject.gtinfosec.org

0

Web Page returned after running the “alert” script

Time to use the Developer Tools!

You can access these tools by going to the three vertical lines in the upper right corner, opening the drop down, going to More Tools → Web Developer Tools. You'll see a new window popup on the bottom half of your screen.



Searching for <script>alert(0)</script>

Your search for returned these results:

No results found.

```
Some cookies are misusing the recommended "SameSite" attribute 2
This page uses the non standard property "zoom". Consider using calc() in the relevant property values, or using "transform" along with "transform-origin: 0 0".
GET https://webproject.gtinfosec.org/favicon.ico
»
```

Web Developer tools opened to show the Console Tab

Useful Default Functionalities: Inspector and Pick an Element

- In our project, two of the most useful tabs will be the “Inspector” tab and “Pick an element” tab (the tab furthest to the left)

- The “Inspector” tab shows the value of various widgets loaded on the page. What sort of information does this reveal about the page? Can you find where your inputs (searches, login credentials, etc.) get loaded?

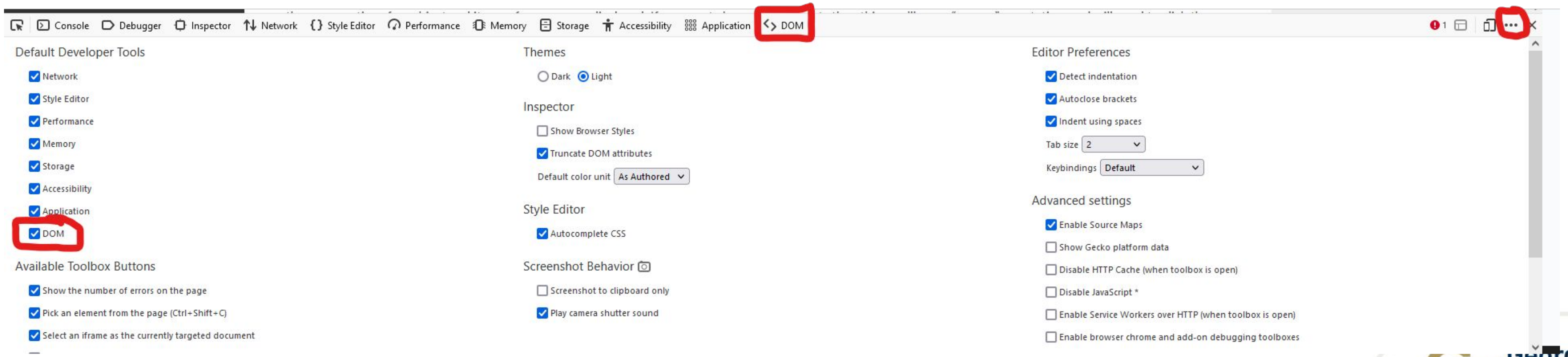
- The “Pick an element” tab operates similarly to the “Inspector” tab, but items on the page are highlighted as you move the cursor around.

- Use these functionalities to understand the elements on the page and figure out how to craft your script.

Georgia Tech.

Accessing DOM Property Viewer

- To get the most out of web developer tools, we need to enable to DOM Property Viewer.
- This is not enabled by default, but it is easy to enable
    - On your web developer console, click the three dots in the upper right corner. A drop down will appear that includes “Settings”.
    - Click “Settings” and several checkboxes under various headings will appear.
    - You should immediately see the “Default Developer Tools” heading.
    - “DOM” will be the last option. Select it and a new tab will be added to your web developer console.



Default Developer Tools
- [✓] Network
- [✓] Style Editor
- [✓] Performance
- [✓] Memory
- [✓] Storage
- [✓] Accessibility
- [✓] Application
- [✓] DOM

Available Toolbox Buttons
- [✓] Show the number of errors on the page
- [✓] Pick an element from the page (Ctrl+Shift+C)
- [✓] Select an iframe as the currently targeted document

Themes
- ○ Dark ● Light

Inspector
- [ ] Show Browser Styles
- [✓] Truncate DOM attributes
- Default color unit: As Authored

Style Editor
- [✓] Autocomplete CSS

Screenshot Behavior
- [ ] Screenshot to clipboard only
- [✓] Play camera shutter sound

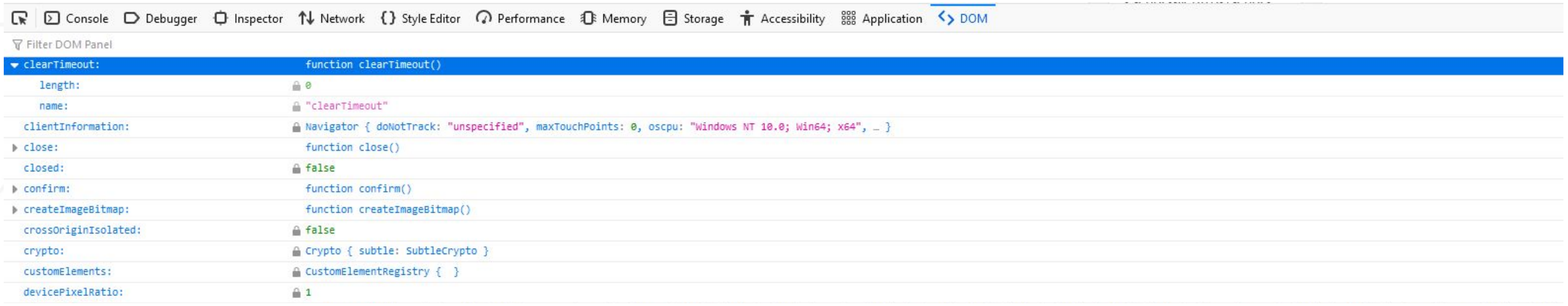
Editor Preferences
- [✓] Detect indentation
- [✓] Autoclose brackets
- [✓] Indent using spaces
- Tab size: 2
- Keybindings: Default

Advanced settings
- [✓] Enable Source Maps
- [ ] Show Gecko platform data
- [ ] Disable HTTP Cache (when toolbox is open)
- [ ] Disable JavaScript *
- [ ] Enable Service Workers over HTTP (when toolbox is open)
- [ ] Enable browser chrome and add-on debugging toolboxes

Georgia Tech.

Functionality of DOM Property Viewer

- Here is one more way to view the elements and their contents on the page. How do the values of elements change with different inputs?



```
▾ clearTimeout:              function clearTimeout()
    length:                  0
    name:                    "clearTimeout"
  clientInformation:         Navigator { doNotTrack: "unspecified", maxTouchPoints: 0, oscpu: "Windows NT 10.0; Win64; x64", … }
▸ close:                     function close()
  closed:                    false
▸ confirm:                   function confirm()
▸ createImageBitmap:         function createImageBitmap()
  crossOriginIsolated:       false
  crypto:                    Crypto { subtle: SubtleCrypto }
  customElements:            CustomElementRegistry {  }
  devicePixelRatio:          1
```

Elements shown on the DOM Property Viewer

XSS Tasks and Tips

- Goal: Construct a URL that executes a malicious payload when it is loaded into the victim's Firefox browser.
- **Important!**
  - None of your scripts should open a separate tab when executed.
  - The page smoothly transition the user to the attacker controlled page after they hit "Search"
- 2.0: No Defenses

  Hint 1: Look into the window.onload event in Javascript. How can it be used?
- 2.1: Remove "script"

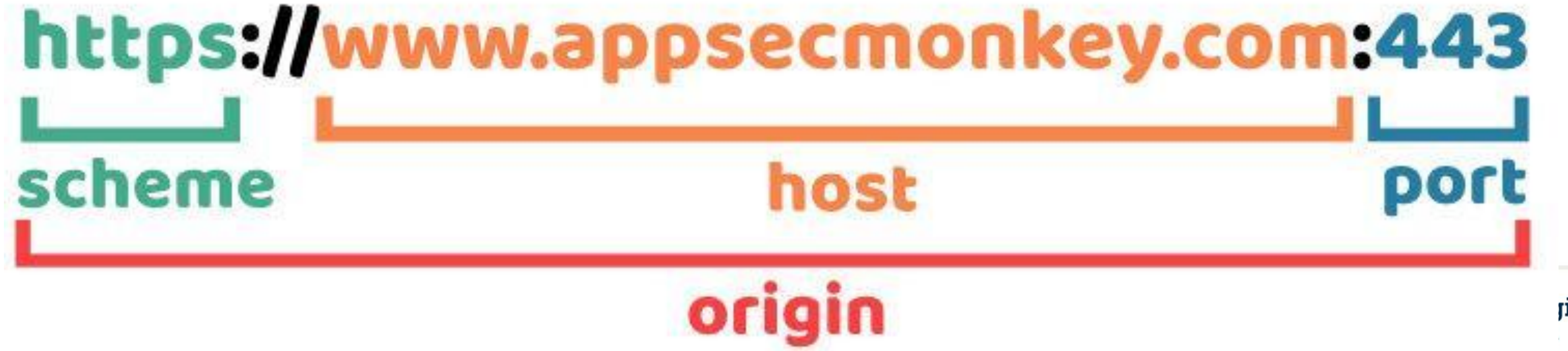
  Hint 2: Look how far a bit of sanitization takes your input!
- 2.2: Remove several tags

  Hint 3: The server won't sanitize input that is within a certain "frame" of reference.
- 2.3: Remove some punctuation

  Hint 4: Javascript has some unique string syntax. Which one is not being sanitized out of the input?

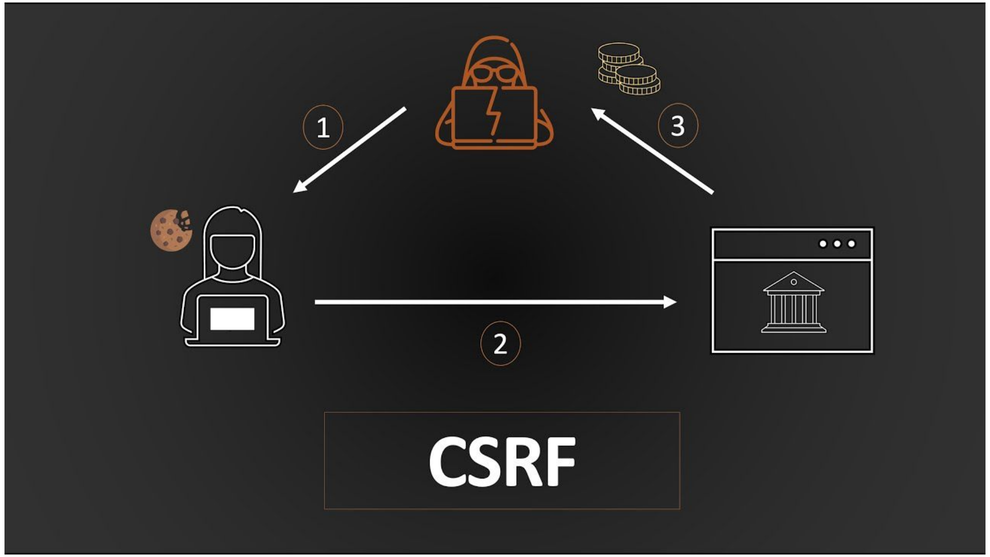
CSRF – Same Origin Policy

- A browser security feature that restricts how resources on one origin interact with those on other origins.
- Checks for same scheme, domain, and port
- SOP prevents reading cross-origin requests, not sending requests

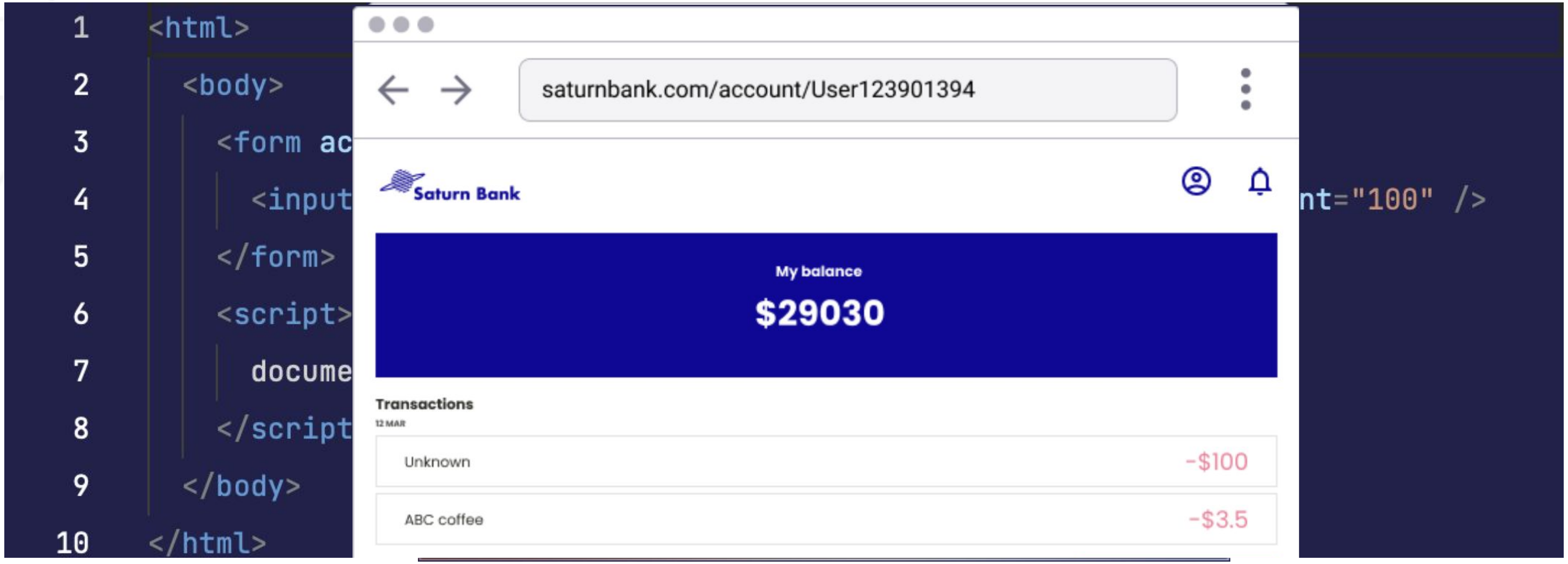


What is CSRF?

- Cross-Site Request Forgery
- An attack that causes a user's browser to submit an unintended request to a web application where the user is already authenticated.
- Server treats the request as legitimate because it includes valid session cookies.



CSRF Attack – Simple Walkthrough



```
<html>
  <body>
    <form ac
      <input                                                  nt="100" />
    </form>
    <script>
      docume
    </script
  </body>
</html>
```

saturnbank.com/account/User123901394

My balance

$29030

Transactions

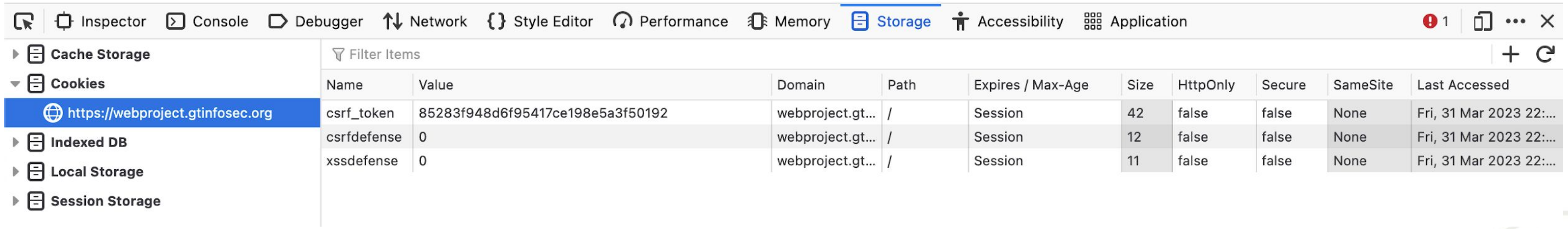
12 MAR

Unknown	-$100
ABC coffee	-$3.5

- Fun Fact – TikTok was affected by CSRF and XSS vulnerabilities in 2020!

CSRF Tokens

- Secure random tokens generated by server during an HTTP browsing session.
- Helps server ascertain that the request originates from a legitimate source.
- Can be found in developer tools under storage
- Tip: Look into how the jQuery ajax function can be used to generate HTTP request to the server



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
csrf_token	85283f948d6f95417ce198e5a3f50192	webproject.gt...	/	Session	42	false	false	None	Fri, 31 Mar 2023 22:...
csrfdefense	0	webproject.gt...	/	Session	12	false	false	None	Fri, 31 Mar 2023 22:...
xssdefense	0	webproject.gt...	/	Session	11	false	false	None	Fri, 31 Mar 2023 22:...

CSRF Attack Tasks

- Goal – Build HTML page that on load would login the victim as ‘attacker’
- Should be a one click attack. Page can be blank.
- Task 3.0 – No Defenses:
  - Login form does not use any CSRF token validation
- Task 3.1 – Token Validation:
  - Server sets a CSRF token hidden on form submit
  - Hint: Try to implement the XSS search vulnerability to obtain the CSRF token and forge the login request. (Remember CSRF token is random and cannot be hard-coded)