



PA4 Part 2: Injection Boogaloo

Patrick Liu, with resources from Riley Hadden



Minor Clarifications on PA4

- Groups only need to complete the assignment on one account
- You cannot go back to previous challenges
 - Contact us to reset your progress if you would like-this is all or nothing!

Base64 Encoding

- Base64 is a method of binary to text encoding
- Allows embedding of arbitrary data in text channels
- Every 6 bits corresponds to a digit
 - {A-Z, a-z, 0-9, +, /}. 62 digits are the same among most implementations, last two digits vary
 - Padded with =
- Use the base64 module in Python to handle this for you

Input String:

W

O

W

Binary:

01110111

01101111

01110111

B64 Grouping:

011101

110110

111101

110111

B64 encoded:

d

2

9

3

Base64 in Python

- Encoding

```
import base64

# Encoding
body = "wow"
body_ascii = [ord(x) for x in body]
body_bytes = bytes(body_ascii)
body_b64_encoded = base64.b64encode(body_bytes)
body_b64_encoded_string = "".join([chr(x) for x in body_b64_encoded])
```

- Decoding
 - `base64.b64decode()`
- [Python docs reference](#)

Cookies

- Strings that websites can request that the browser store
- Cookies are sent by the browser along with requests
 - Websites decide how to respond to cookies
- Typically used to serve user-specific content(e.g if a user is logged in)



Cookies cont'd

- Cookie same-origin policy: ([scheme], domain, path)
 - `scheme://domain:port/path?params`
- Browser sends all cookies that match:
 - Scheme
 - Domain suffix
 - Path prefix
- Why might we not want website A to see website B's cookies?

Cross-Site Request Forgery(CSRF)

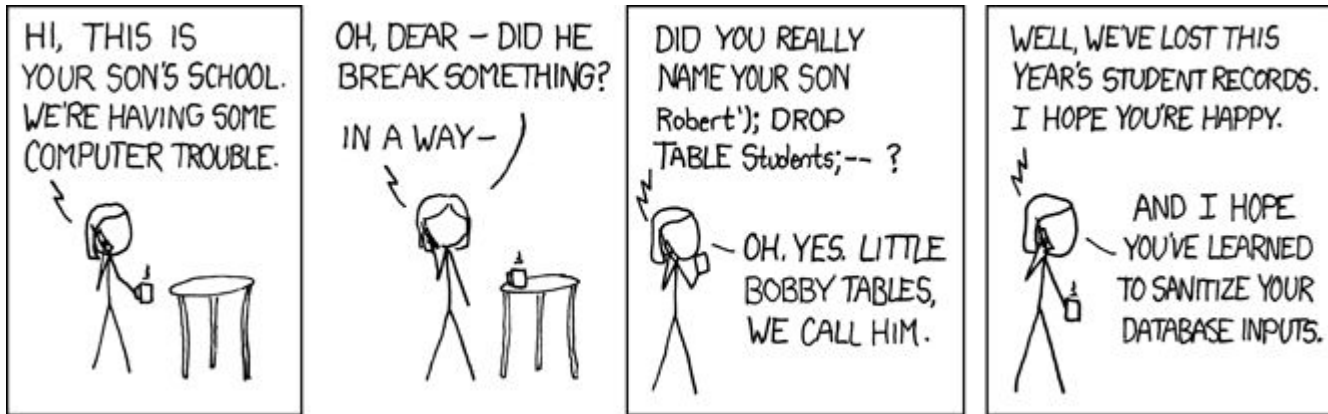
- Evil site can make request to legitimate site
 - Evil.com might make a request to bank.com
- Browser will send cookies for bank.com with request
 - Evil.com can't inspect the response due to SOP, does it matter?
 - Can manipulate state in bank.com!
- Defenses
 - CSRF token: Bank.com pages will know the token, evil.com will not
 - Referer/Origin header: Informs server who made the request
 - SameSite cookies

SQL Injection

- Suppose we have the query:

```
Select * FROM students WHERE username='${username}' AND  
password='${password}'
```

- How might you make this query select a student without knowing their password?
- How might you make this query modify the table?
- SQL reference: https://www.w3schools.com/sql/sql_quickref.asp
 - May be helpful for challenge 7/8



- What does this comic get wrong?

SQL Injection Prevention

- Prepared statements/Object-Relational Mappers(ORMs)

```
sql = "SELECT * FROM users WHERE email = ?"
```

```
cursor.execute(sql, ['ptliu@ucsd.edu'])
```

- Do not try to sanitize

Javascript Injection

Source

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>This is a Heading</h1>
<p>This is a paragraph with some user info: {}</p>
```

- Suppose {} takes user input
 - What happens if we supply Javascript?

Rendering

This is a Heading

This is a paragraph with some user info: {}

Source

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>This is a Heading</h1>
<p>This is a paragraph with some user info: <script>document.write("\tHello, from inside the
script!")</script></p>
```

Rendering

This is a Heading

This is a paragraph with some user info: Hello, from inside the script!

Cross-Site Scripting(XSS)

- If we can embed the previous script into page HTML, we can have it execute in the user's browser.
- Stored XSS:
 - Script that's stored in a database, then displayed to users later
 - E.g Forum signatures
- Reflected XSS:
 - Some websites will reflect query strings in the URL on the page
 - <https://duckduckgo.com/?q=xss>
 - One can imagine HTML of the form `<input value="...>`
 - Query string of form `"><script>alert("XSS")</script>`
 - Script runs with permissions of page!

XSS Prevention

- Sanitize inputs
 - Not easy! Suppose we filter `<script>` tags. What if we encode "<" as %3C?
 - What if we want users to be able to do stuff with scripts?
- Content Security Policy(CSP)
 - Browser will only fetch content or execute scripts from whitelisted domains
 - Served via HTTP Header or embedded in the page