CS3235 Tutorial 10

SQL injection and Cross-Site Scripting

Threat Model

CSRF defences

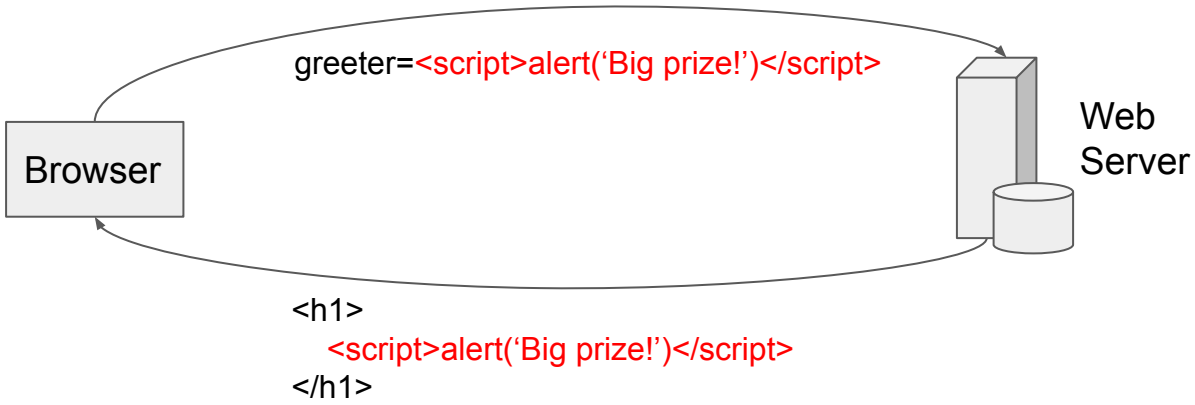
1. Tokens
2. Samesite cookies
3. Referer

How to defeat those defences?

Idea: trigger a malicious request from the same origin

XSS
Cross-Site Scripting

Reflected XSS



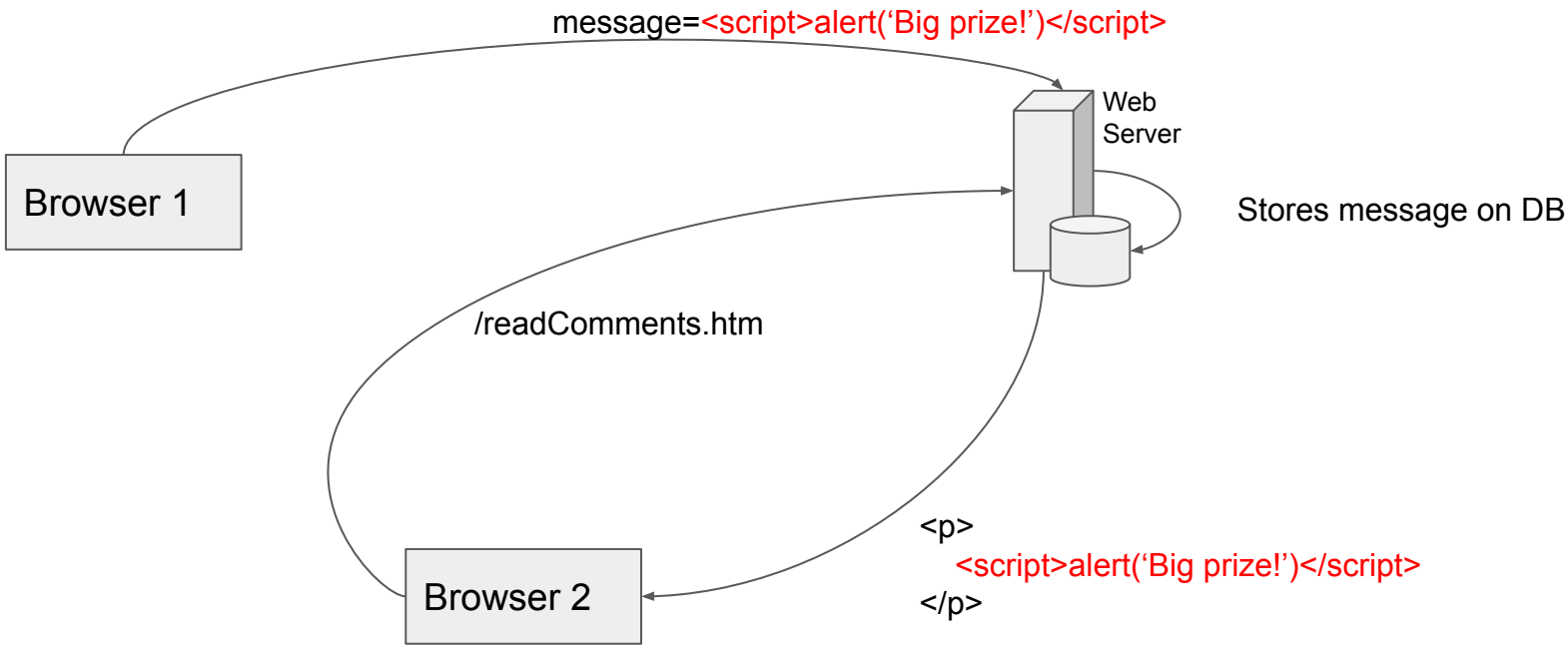
Bad things can happen

What can happen?

- Tamper with content on page
- Steal data
  - Cookies
  - CSRF tokens
- Forge requests on the user’s behalf

Convince/lure the **victim to visit the malicious URL**

Stored/Persistent XSS

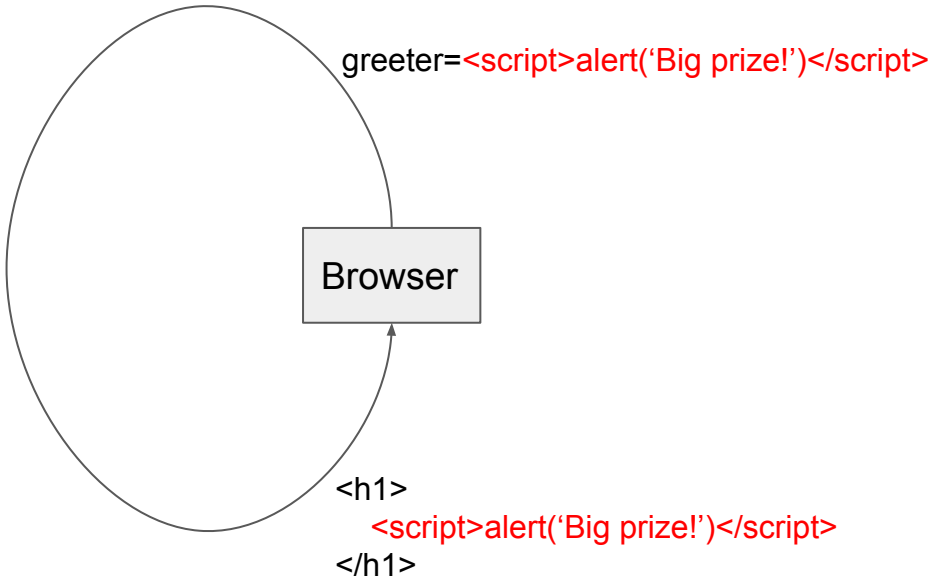



Twitch chat and exploits: Destroying dwangoAC's IRC client

Stored XSS

Reflected XSS

DOM-based XSS



How to spot?

- Look for webserver responses…
  - see if it responds something that is identical to a user input
- Values read from resources that can be access by third party website:
  - XMLHttpRequests
  - Form Inputs
- Try to store some data on the webserver and read it back,
  - see if it has changed
  - try storing script in cookies, forms, etc

Defences

CSRF defences?

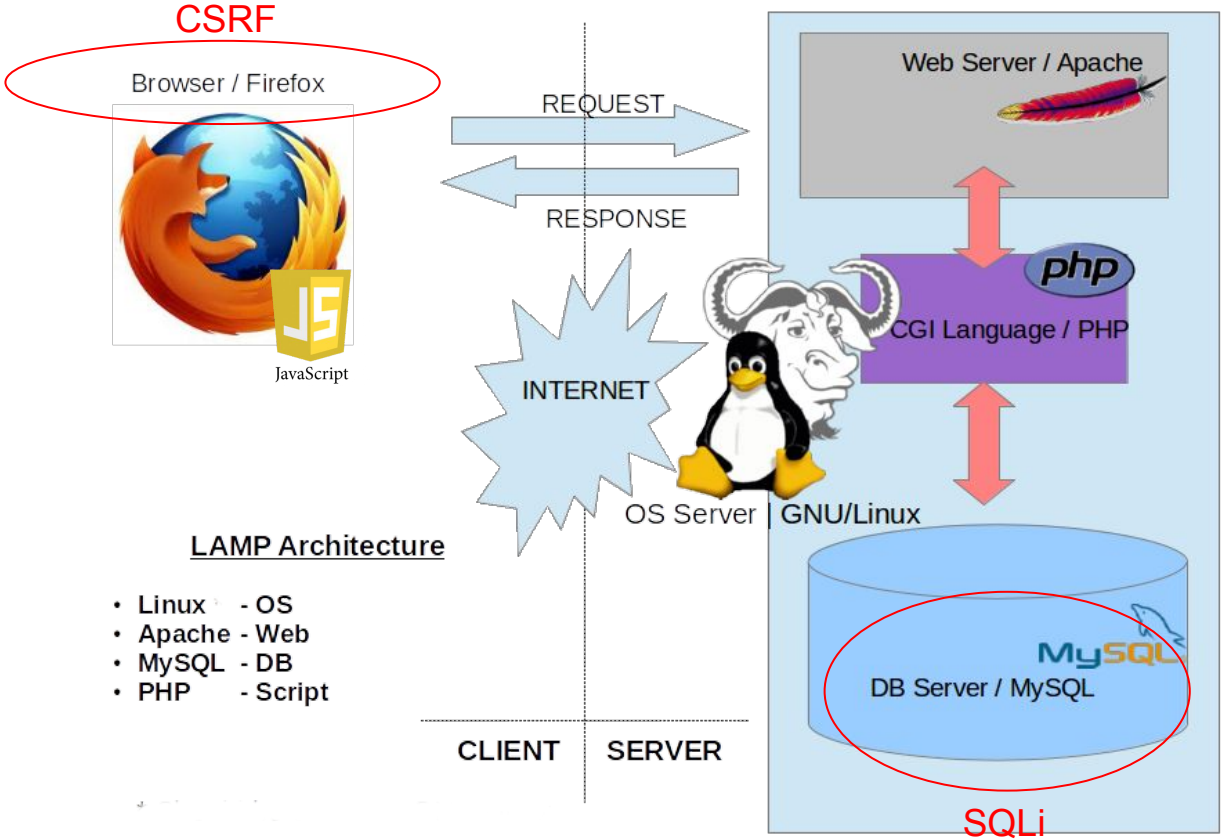
- CSRF tokens?
- Samesite cookies?
- Referer?

Defences

- Server-side: reflected and stored
- Browser-side: DOM
- Block <script>?
  - There are other ways: event handlers
  - <svg onload = “alert(1337)”>
- Block “<” entirely?
  - Email services use html to send email
  - Rich text editors use them
  - Online website builders use them
- Libraries that **sanitise** input to mitigate XSS risks..
  - DOMPurify

Recap: Web Primer

Lamp:



If exploited,

- The attacker can:
  - View data that it normally cannot (including other users' data)
  - Modify/Delete the data permanently
  - In some cases: escalate the attack to compromise the underlying server or other servers in the network.
  - Denial of service

  … pretty scary right?

SQL Primer

```
SELECT * FROM products
WHERE category = ‘Gifts’
AND released = 1 --Check if the prod is released
```

What does this query do? (apart from shouting)

Demo

https://www.hacksplaining.com/exercises/sql-injection

Examples of SQL Injections

https://whomakesawebsitelikethisanyways.com/products?category=Gifts

```
SELECT * FROM products
WHERE category = 'Gifts'
AND released = 1
```

https://whomakesawebsitelikethisanyways.com/products?category=Gifts'--

```
SELECT * FROM products
WHERE category = 'Gifts'--'
AND released = 1
```

Examples of SQL Injections

https://wellokcomeonreally.com/products?category=Gifts

```
SELECT * FROM products
WHERE category = 'Gifts'
AND released = 1
```

https://wellokcomeonreally.com/products?category=Gifts'+OR+1=1--

```
SELECT * FROM products
WHERE category = 'Gifts' OR 1=1--'
AND released = 1
```

Examples of SQL Injections

https://wellokcomeonreally.com/products?category=Gifts

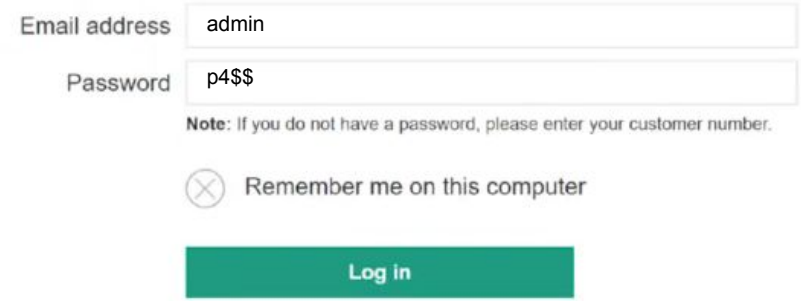
```
SELECT * FROM products
WHERE category = 'Gifts'
AND released = 1
```

https://wellokcomeonreally.com/products?category=Gifts' UNION SELECT * FROM users--

```
SELECT * FROM products
WHERE category = 'Gifts' UNION SELECT * FROM users--
AND released = 1
```

Examples of SQL Injections

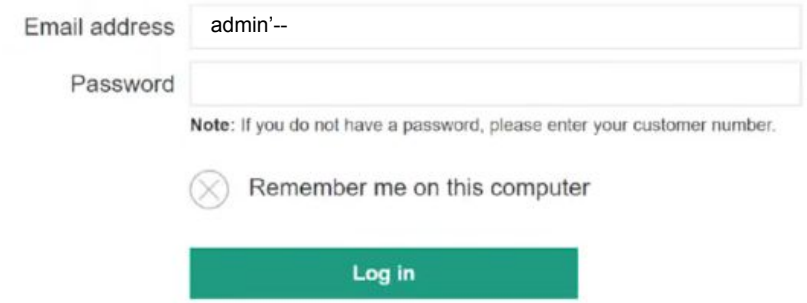
https://paysia.com/login




```
SELECT * FROM users
WHERE uname = 'admin'
AND pass = 'p4$$'
```

Examples of SQL Injections

https://paysia.com/login




```
SELECT * FROM users
WHERE uname = 'admin' --'
AND pass = ''
```

You get the idea!

- `SELECT * FROM information_schema.tables`
- `SELECT * FROM v?version (Oracle)`
- `SELECT * FROM items WHERE (price/0)>0`
- `SELECT * FROM items AND WAITFOR DELAY(60000)`

Check out the cheat sheet:

https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/

WHERE does the VULNERABILITY occur?

- WHERE clause of almost any kind of query
- UPDATE statement: Values
- INSERT statement: Values
- SELECT statement: Table or Column name
- ORDER BY clause: SELECT statement
- etc.

Injection can happen anywhere

```
bash -c “echo $1”

dd if=$1
```

Supplementary Materials

https://www.hacksplaining.com/lessons

https://owasp.org/www-community/attacks/DOM_Based_XSS

https://cure53.de/fp170.pdf