UBC LAUNCH PAD

Application Security

It’s worth a shot.

https://youtu.be/7W5au-IJUEc

Approach

1. What created the vulnerability.
2. How the vulnerability is exploited.
3. How to protect yourself.

Web 2.0

What could possibly go wrong?!

- Servers send HTML and JS to clients
- Clients execute JS that changes the DOM and makes requests back to the server

Authentication

Username and password... easy.

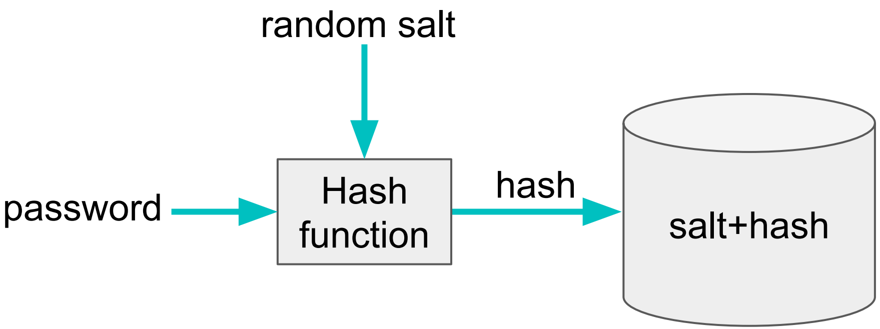
- Usually requires a username and password
- What kinds of passwords are acceptable?
- How should we send the username and password?
- How should we store and validate the username and password?

Authentication

- Don’t store passwords in plaintext
- Hash!

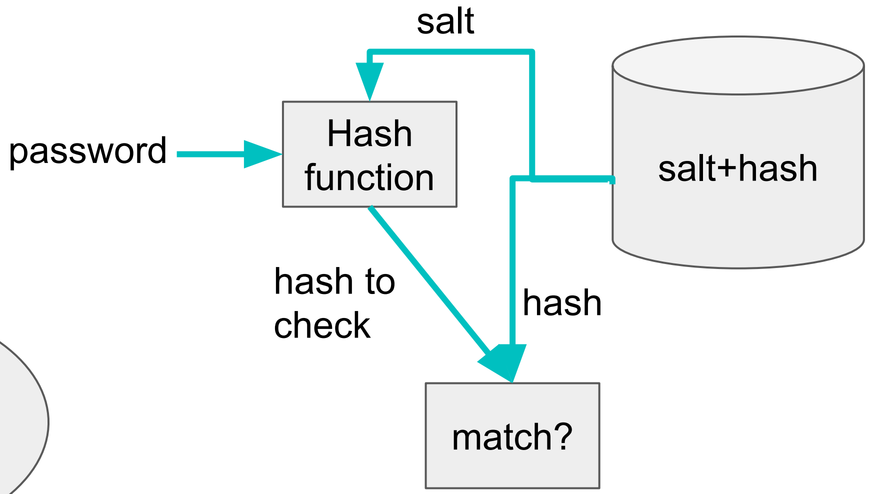
Authentication

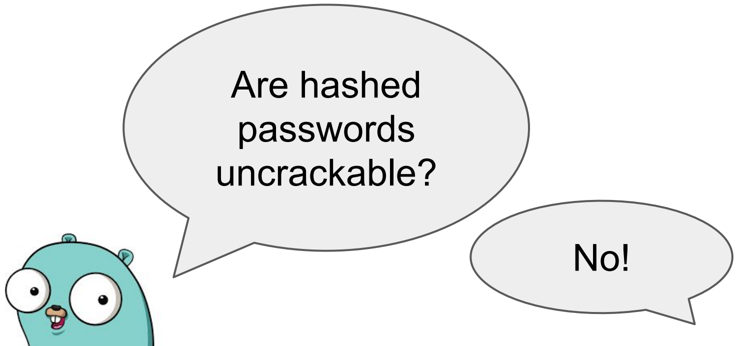
Password Storage



Authentication

Password Validation

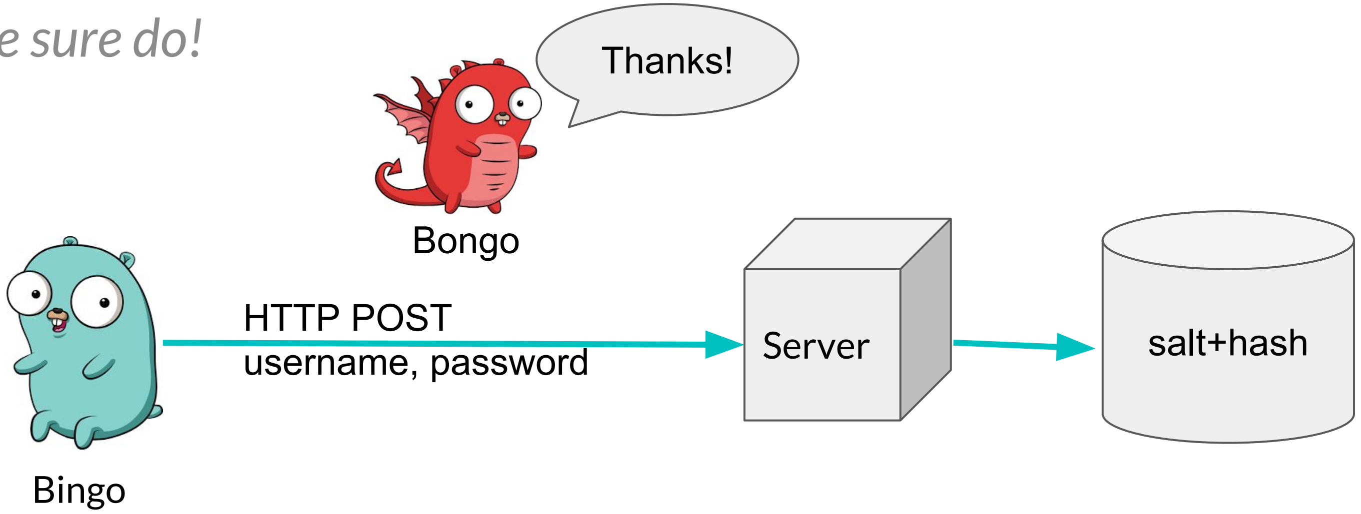




Authentication

Encryption

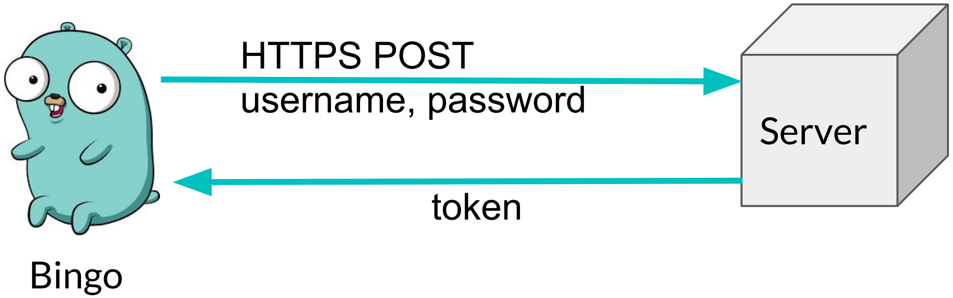
- We're hashing passwords, so do we need encryption?
- *We sure do!*



Authentication

Tokens

- We don’t want people to have to constantly log in
- We need to give the client a token that they can use to prove that they have authenticated successfully

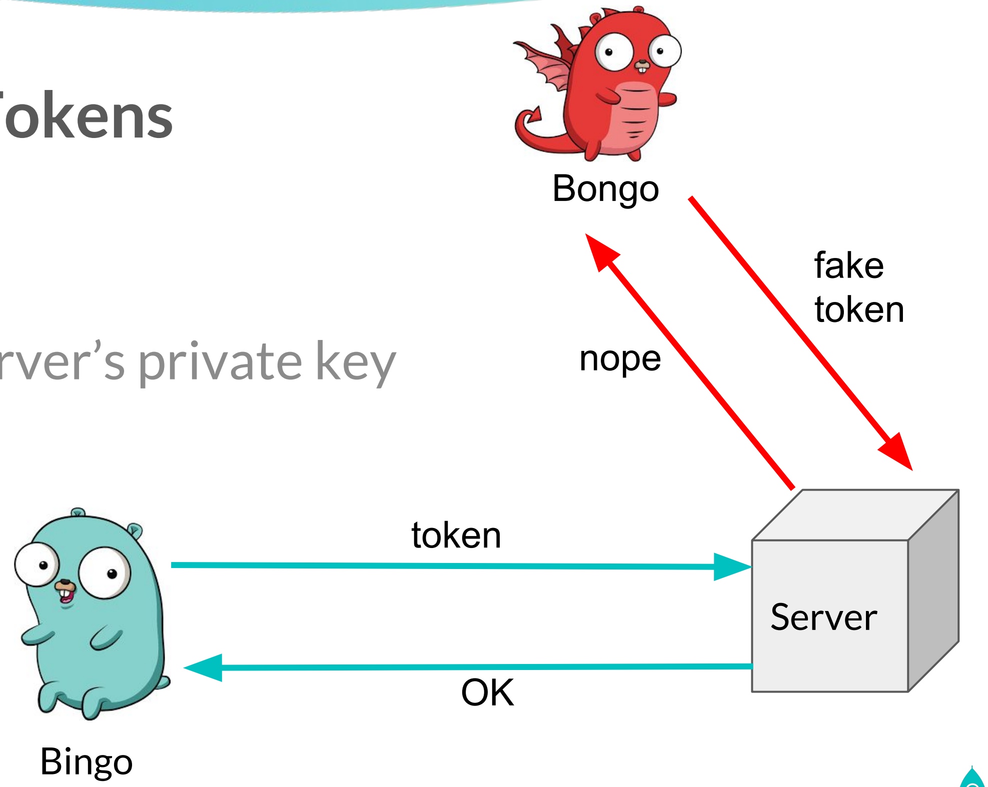


Authentication

Tokens

- Must identify the user
- Must be signed with the server's private key

```
{
    "username": "Bingo"
    "signature": "asdfdieuwr23j98we7fdjk"
}
```



SQL Injection

- What happens when name is `“ or “” = “`

```python
def handle_request(request):
    name = request.params.name
    results = db.execute(
      f'SELECT * FROM users WHERE name = "{name}"')
    return results
```

SQL Injection

Solving the problem.

- Blacklist certain characters
- Whitelist certain characters
- Use prepared statements

```
def handle_request(request):
    name = request.params.name
    query = db.query('SELECT * FROM users WHERE name = ?', name)
    return query.execute()
```

Cross-site Scripting (XSS)

Just stick to the script.

- Stored XSS
- Reflected XSS

www.bongo.com

Check this out!
www.bank.com/profile?name=<script>...

```
def handle_request(request):
    return f'''
    <p>Welcome, {request.params.name}!</p>
    '''
```

Cross-site Scripting (XSS)

Protection.

- Sanitize inputs
- Escape HTML
- Use auto-escaping framework like React or Vue.js

Cookies

Just maintain state! No problem!

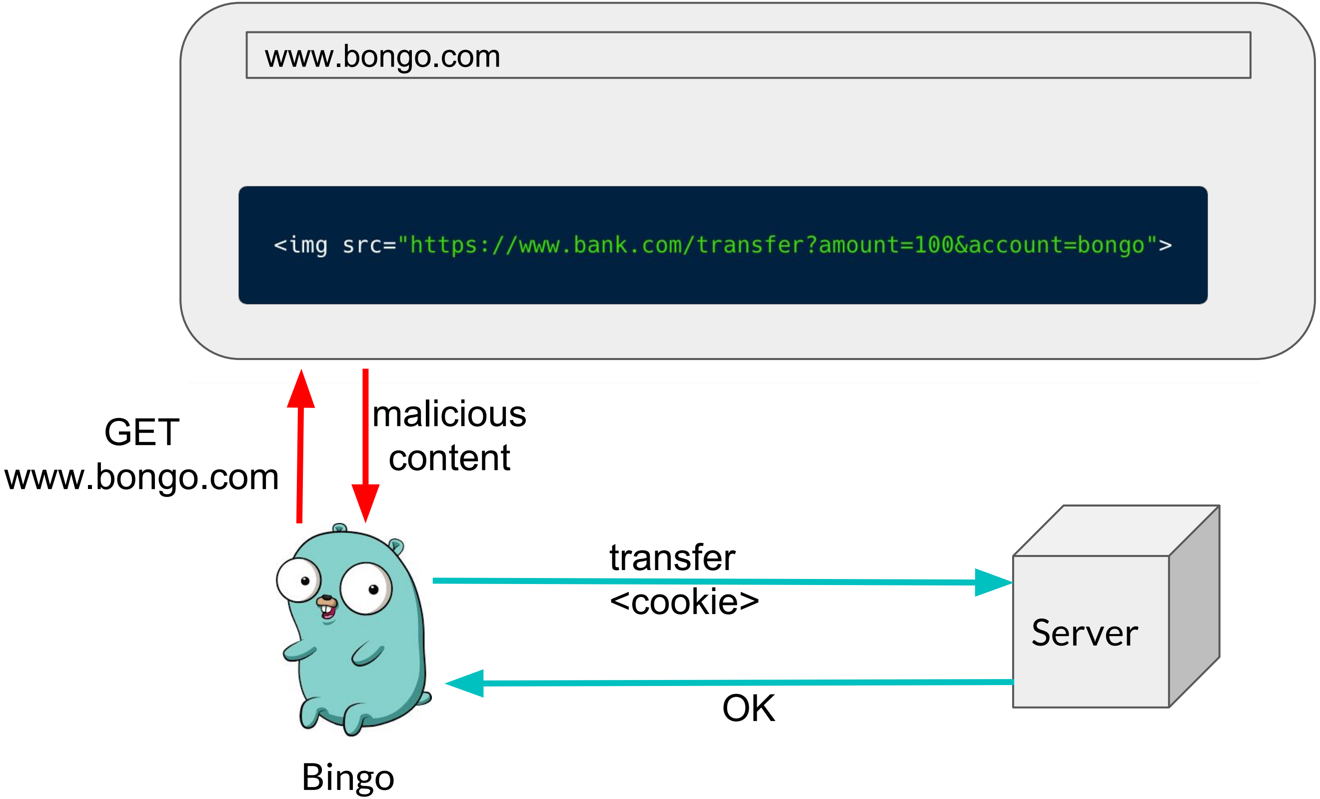
- HTTP is stateless
- Cookies are used to maintain state
  - Store session information
  - Store user preferences
  - Track your every move...

Cross-site Request Forgery (CSRF)

It really wasn't me this time.

- Your browser automatically attaches cookies to requests to the domain they came from

Cross-site Request Forgery (CSRF)



Cross-site Request Forgery (CSRF)

Prevention.

- Don’t use GET to modify state
- Hidden nonces in forms
- Use **Samesite** cookies
- Check the origin or referer of the request

Honorable Mentions

- Containers
- Metasploit (penetration testing): www.metasploit.com
- OWASP (web application security): www.owasp.org
- WebGoat: github.com/WebGoat/WebGoat