Web and API Security

By Cyrus Malekpour

About Me

- Software Developer at 

  - Developing secure webapps and exploiting not-so-secure ones ;)
- Founding member of TJHSST Computer Security Club
  - Founder of CNSUVA

What is Web Security?

- Securing online applications against malicious attackers
- Preventing data capture or exfiltration
- Online applications are increasingly under attack
  - 5 out of 6 large companies were hit last year (Symantec)
  - Attackers are getting more sophisticated

Why is it so common?

1. Most people don’t think about security.
2. Security is hard. Actually, really hard.
3. Nobody understands the whole system

Physical security isn’t perfect either!


Anthem®


Anthem


Amerigroup
RealSolutions
in healthcare


CareMore
It's what we do.™

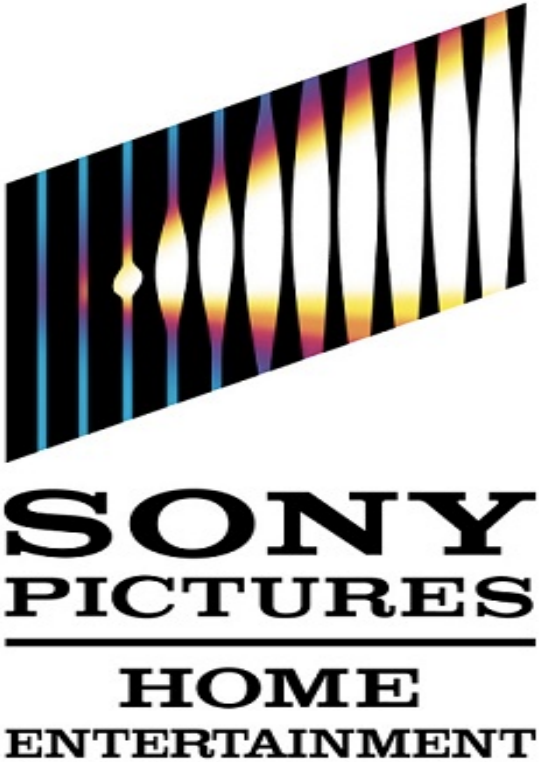

UniCare

Anthem Inc.

- 2nd largest healthcare provider in the US
- **80 million individuals** had their info stolen
  - "names, birthdays, medical IDs/social security numbers, street addresses, email addresses and employment information, including income data"
- Basically **your identity**.

Sony Picture Entertainment

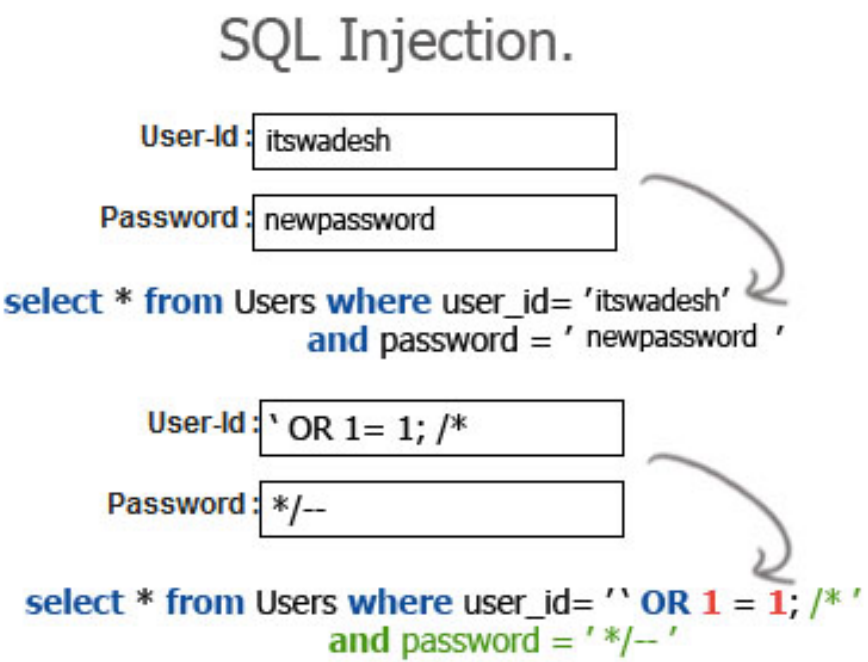
- AKA “How James Franco almost didn’t kill Kim Jong-un”
- **Almost a year** in Sony’s systems
- Tons of internal info, employee info, etc.
- CEO stepped down a month later – coincidence?



OWASP Top 10

#1: Injection

- Malicious input leads to unintended code execution
- Most common example is SQL injection
  - Any interpreter is a target for injection
- Mass Assignment



How to prevent SQL injection

- SQL queries are **unichannel** with the data sent
- node-mysql
  - Placeholders and prepared statements

```
var userId = 'some user provided value';
var sql    = 'SELECT * FROM users WHERE id = ' + connection.escape(userId);
connection.query(sql, function(err, results) {
  // ...
});

/* -- */

connection.query('SELECT * FROM users WHERE id = ?', [userId], function(err, results) {
  // ...
});
```

Mass Assignment

- Common to a lot of large web frameworks
- Assigning fields that weren't meant to be assigned or accessible
  - Not present in forms != not accessible
- Takes advantage of "active record" web app setups
- Data inserted into POST request and passed to database update method

Active Record

Rows in the database are represented as objects

“select * from users where id = 1”

(returns parseable database result)

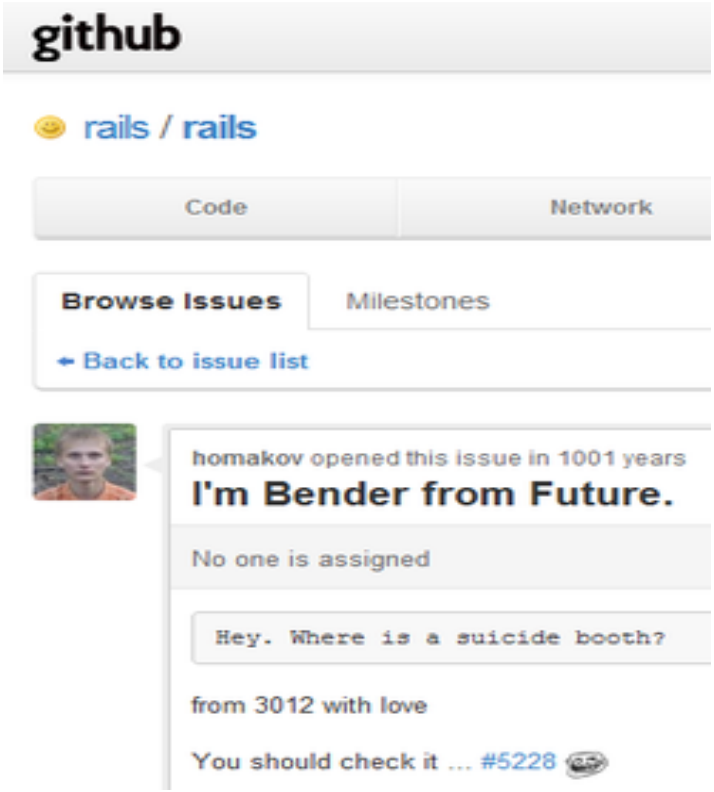
Users.get(1)

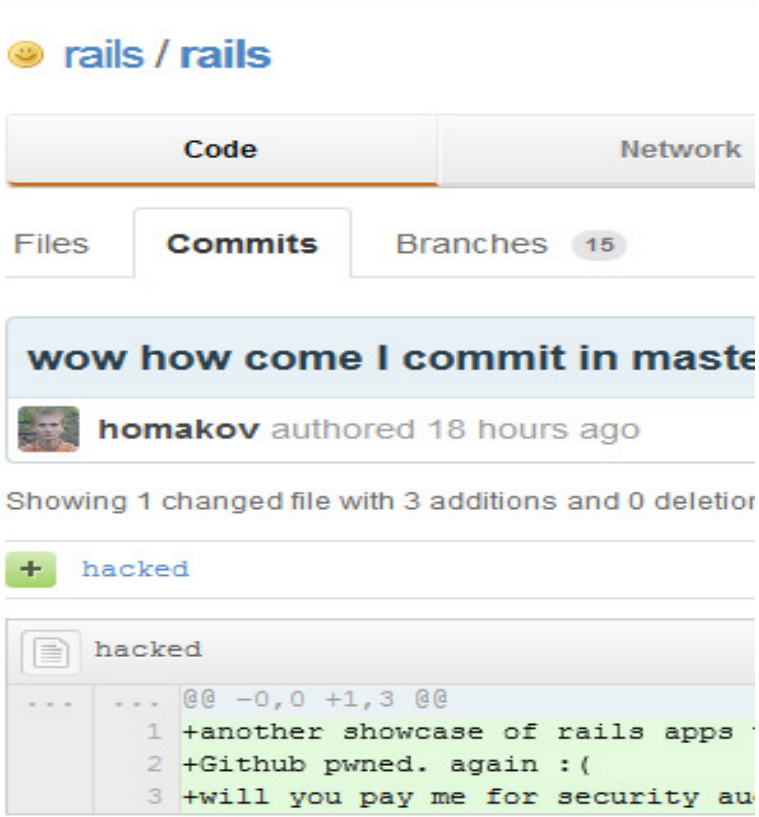
(returns direct user object)

GitHub example

```
class PublicKeyController < ApplicationController
  before_filter :authorize_user
  ...
  def update
    @current_key = PublicKey.find_by_id params[:key]['id']
    @current_key.update_attributes(params[:key])
  end
end
```

GitHub pwned





#2: Broken Auth./Session Mgmt.

- Poor password hashing
- Session fixation attacks
- Improper handling of the session token
  - Storing it in the URL
- Leaking information about existing accounts
  - “Email doesn’t exist” vs. “Incorrect Password”

Password Hashing

- Passwords should be protected even when attackers can access the database
- We only need to **check** passwords, the actual contents are irrelevant!
  - One way functions: hashes
- Hashes should be computationally intensive and impossible to reverse
  - Avoid MD5, SHA1, etc.

#3: Cross-site Scripting (XSS)

- Improperly escaped user input in the browser
- Most prevalent web security flaw
- Malicious input allows attacker to run code on the target's browser

```
<html>
    <head>
        <title>Hello World!</title>
    </head>
    <body>
        My name is <?php $_GET['name']; ?>
    </body>
</html>

Input: "<script>alert('attacked')</script>"
```

Different types of XSS

- Reflected
  - Never touches server, passed in from URL
- Stored
  - Retrieved from server and put directly into page
- HTML vs. JavaScript

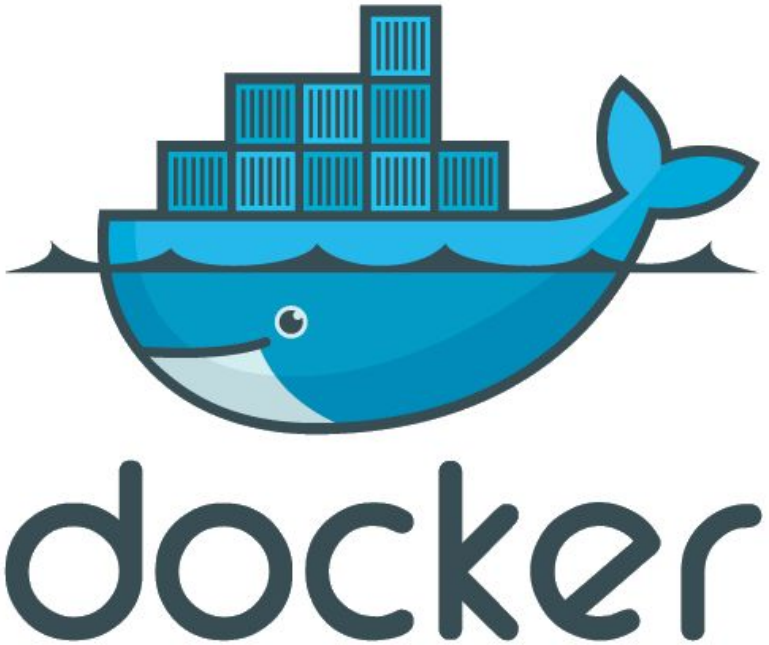
```
# controller
@attrs = {:email => 'some@email.com</script><script>alert(document.domain)//'}

<!-- view -->
<script>
  var attributes = <%= @attrs.to_json %>
</script>
```

#4: Insecure Direct Object Reference

- Application doesn’t verify the user is able to access resources
- Merely hiding links or pages doesn’t prevent users from directly navigating to them

#5: Security Misconfiguration

- Software deployment is extremely important for security
- Simple to address but easy to overlook
- Rise of “devops” has lead to consideration of production system config during development



#6: Sensitive Data Exposure

- Browser security headers
- Use HTTPS whenever possible
  - HSTS
  - Weak cryptography
- Server or client caching
- “Where do our logs go?”

#7: Missing Function Level ACL

- “Security through obscurity”
- No protection from the user directly accessing a page
- Client side checks are only for quick **validation**, not security!

#8: Cross-site Request Forgery (CSRF)

- Forcing authenticated users to do something unintended
- Request is seen is valid, so difficult to detect
- User must be tricked into making request
  - Cannot perform attack without user's direct action

Example

http://bank.com/transfer.do?acct=BOB&amount=100

http://bank.com/transfer.do?acct=MARIA&amount=100000

#9: Components w/ known Vulns

- Outdated dependencies often have well known public vulnerabilities
- Keeping track of library versions used by the application is crucial
- npm makes this easier
  - *package.json*

#10: Insecure Redirects

- Unchecked redirects can lead users to malicious websites
- Common pattern is “redirect on login”

Example exploit URL

http://www.example.com/redirect?url=evil.com

Useful Tools

- SSL Labs
  - https://www.ssllabs.com/
- Node Security
  - https://nodesecurity.io/
- *npm shrinkwrap*
  - Bundled w/ npm
- retire.js
  - Helps find modules with known vulnerabilities
- helmetjs (for express)
  - Adds some security-related HTTP headers

Key Points

- In the real world, you **cannot trust the user**
  - Assume any input source can lead to an attack
- Defense in depth
  - Check on the client, then server, then on output, etc.
- Client-side checking is *strictly* for quick validation
  - Only meant to enhance the user experiences
- Good security doesn't need to compromise UX

Questions?