Web Application Security 101

Kip Robinson

NCDevCon 2015

Who I Am

- Developer at Verian
- Graduate of NCSU (go Wolfpack!)
- Taught myself to code on a TI-83
- 11+ years experience
- Things I've worked with: C++, Java, ColdFusion, PHP, Perl, MySQL

Overview

- Basic security best practices that ***every*** developer should know
- Security is hard… but security basics are not!
- "I am not a security expert" is ***not*** an excuse

Overview

- Deep Dive on OWASP Top 4
- Hacking a poorly-written application (how-to!)
- Only try this at home!

SQL Injection

OWASP #1

SQL Injection

- Happens when user-submitted values are inserted directly into SQL query

- Malicious user can modify query to:
  - Return unauthorized data
  - Gain unauthorized access
  - Destroy data

SQL Injection Example

Query in our code during login:

```
SELECT *
FROM Users
WHERE username = '#Form.un#'
AND   password = '#Form.pw#'
```

SQL Injection Example

Query in our code during login:

```
SELECT *
FROM Users
WHERE username = '#Form.un#'
AND   password = '#Form.pw#'
```

User submits:

```
un:  admin' UNION SELECT * FROM Users WHERE username = '
pw:  123
```

SQL Injection Example

Query in our code during login:

```
    SELECT *
    FROM Users
    WHERE username = 'admin' UNION SELECT * FROM
    Users WHERE username = ''
    AND   password = '123'
```

User submits:

```
un:  admin' UNION SELECT * FROM Users WHERE username = '
pw:  123
```

SQL Injection Example

Query in our code during login:

```
SELECT *
FROM Users
WHERE username = 'admin'

UNION

SELECT *
FROM Users
WHERE username = ''
AND   password = '123'
```

SQL Injection

- Other examples:

```
' OR '1'='1

'; DROP TABLE MedicalRecords; --
```

SQL Injection Prevention

- What **NOT** to do:
  - Blacklist problematic characters (single-quote, backtick, etc)
  - Blacklist problematic words (DROP, ALTER, etc)
  - Escape special characters manually
    - Database settings can change escaping methods
    - Can work if you fully control database, but there are better ways

SQL Injection Prevention

- ColdFusion: <cfqueryparam>
- ColdFusion (cfscript): query.addParam()
- Other languages:
  - Google: how to use prepared statement in *language*
- Use abstraction layer / ORM
  - These don't always prevent injection!
  - NoSQL is not a silver bullet!

Other Injections

- Same principles apply when including user-provided data in any command:
  - Example: <cfexecute> to run console commands: generatePdf.exe -input USERDATA.xml && del c:\inetpub\wwwroot\web.config

- Or when loading files:
  - Example: downloadAttachment.cfm?filename=myimg.gif
  - User calls filename=c:\coldfusion11\cfusion\lib\datasources.xml

Bad Session Management

OWASP #2

Session Management

- Internet is **stateless**
- Browser sends session identifier to server on every request to appear stateful
- If attacker gains Session ID, they can hijack session

Session Management

- Session ID Cookies:
  - ColdFusion: CFID, CFTOKEN (or JSESSIONID)
  - Java: JSESSIONID
  - PHP: PHPSESSID

- Treat these as if they are passwords!
  - Never include in URL or in email content

Session Management

- ColdFusion: Server Settings > Memory Variables > Session Cookie Settings:
    - HTTP Only: browsers (modern browsers and IE6+) do not allow JavaScript code to read HTTP-Only cookies
    - Secure Cookie: Only sets session cookie over HTTPS connection
        - If you need to use session, you need to use HTTPS!

Session Management

- <cflocation> - always use addToken="false"
  - Default is addToken="true" for no good reason

Cross-Site Scripting (XSS)

OWASP #3

Cross Site Scripting (XSS)

- User-provided data is used directly in markup/Javascript
- Similar to SQL Injection
  - “Cross Site Scripting” is (IMHO) a terrible description
  - Better names might be:
    - **HTML Injection**
    - **DOM Injection**
    - **Javascript Injection**

XSS Example

- Un-escaped default value for input field

```
<input name="un" value="#qUsr.Name#" />
```

XSS Example

- Un-escaped default value for input field

```
<input name="un" value="#qUsr.Name#" />
```

- User sets username to:

```
jdoe" onfocus="alert('hello!')
```

XSS Example

- Result:

```
<input name="un" value="jdoe"
onfocus="alert('hello!')" />
```

XSS Example

- alert() is just annoying…

XSS Example

- alert() is just annoying…

- More malicious:

```
onfocus=
document.getElementById('userId').value=1;
document.getElementById('password').value='123';
document.getElementById('myForm').submit();
```

Another XSS Example

- Un-escaped value in HTML

```
<li>#qUsr.Name#<li>
```

Another XSS Example

- Un-escaped value in HTML

```
<li>#qUsr.Name#<li>
```

- User sets username to:

```
jdoe<script>alert('hello!')</script>
```

XSS Prevention

- ColdFusion (10+) added ESAPI library:
  - `encodeForHtml()`
  - `encodeForHtmlAttribute()`
  - `encodeForJavascript()`
- Older ColdFusion: HtmlEditFormat(): escapes HTML special characters:
  - `&` → `&amp;`
  - `<` → `&lt;`
  - `>` → `&gt;`
  - `"` → `&quot;`
  - `'` → `&#x27;`
- PHP: `htmlspecialcharacters()`
- jQuery: use `.text()` instead of `.html()`

XSS Prevention

- If you need to support some markup in untrusted data:
  - Use plaintext format like Markdown
  - Use OWASP Antisamy to sanitize HTML input

Direct Object References

OWASP #4

Direct Object References

- Object reference (e.g. database id) is passed in clear

- System assumes this is a safe value

Direct Object References - Examples

- editUser.cfm?userId=3442

- <input type="hidden" name="userId" value="3442" />

Direct Object References - Prevention

- Check permissions when form is submitted
  - editUser.cfm – if URL.userId != session.userId, and user does not have “admin” role, return authorization error

  - This must be a habit for all developers, all the time!

Direct Object References - Prevention

- Encrypt data passed in URLs or hidden form fields
  - You must know what you are doing!
  - If encryption is compromised, you have a false sense of security

Direct Object References - Prevention

- Obfuscate IDs – instead of sequential IDs, use (for example) random 8-character alphanumeric ID
  - Makes guessing another ID difficult
  - Requires more work when inserting data
  - Still not helpful if obfuscated ID is exposed

Session Hijacking with XSS

Live Demo

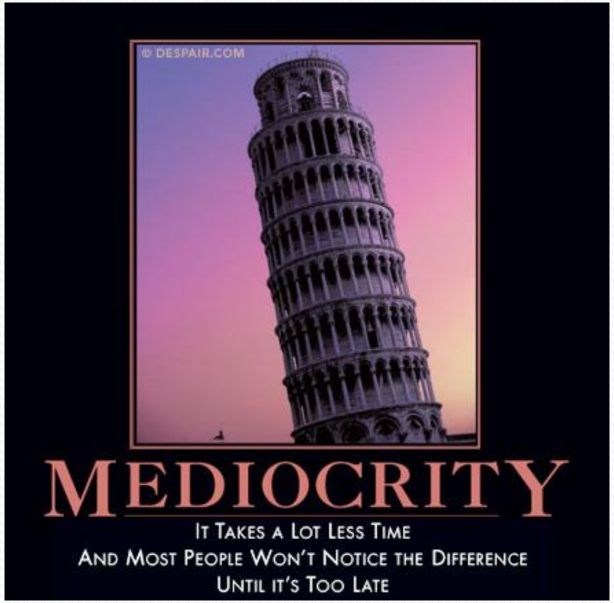
Incentivizing Secure Code

Incentivizing Secure Code

- ***Why*** do developers write insecure code?
    - Sometimes ignorance and/or incompetence

Incentivizing Secure Code

- ***Why*** do developers write insecure code?
  - **Secure coding is not often incentivized**



Incentivizing Secure Code

- **Teach <u>Developers</u>** how to write secure code!
  - Everyone has to learn somewhere
  - Basics are not as hard as it seems

  - Constructive code reviews
    - Especially for less experienced developers

  - Internal training sessions
    - Not everyone comes to developer conferences!

Incentivizing Secure Code

- **Teach** **<u>QA</u>** how to test security!
  - Code snippets to put in form fields to check for XSS
  - Use browser tools to modify hidden/disabled fields
  - Watch for object IDs in URLs. Modify them if found.
  - Watch for AJAX calls in Network tab.

Incentivizing Secure Code

- **<u>Management</u>** must be committed
- More secure code = more time
  - Additional Development Time
  - Internal training time
  - QA Time
  - Code Review Time
- **More time = more cost**

Final Questions?

Slides and code will be posted on:

http://ampersand.space

https://github.com/kiprobinson

Twitter: @kipthegreat

Resources

- OWASP: https://owasp.org
- OWASP Top 10: https://www.owasp.org/index.php/Top_10_2013-Top_10
- ColdFusion Lockdown Guide: http://www.adobe.com/content/dam/Adobe/en/products/coldfusion/pdfs/cf11/cf11-lockdown-guide.pdf