Writing Secure CFML

Pete Freitag, Foundeo Inc.

foundeo

foundeo.com | hackmycf.com | fuseguard.com

About Me

- 15 Years ColdFusion Experience
- 7 Years Foundeo Inc. Consulting & Products
    - cf.Objective() Sponsor
- blog: petefreitag.com
- twitter: @pfreitag

Agenda

- File Upload Vulnerabilities
- Path Traversals
- Cross Site Scripting
- Authentication
- Encryption
- SQL Injection (in 60 Seconds)

File Uploads

Proceed with caution

(cc) http://www.flickr.com/photos/zigazou76/3702501888/

File Uploads Rule #1

Never trust a MIME type

Never trust a MIME

- CF9 and below use the MIME type passed by the browser / client.
  - Hacker can send any MIME type.
- CF10 does a server side file inspection (when strict=true, default).
  - We can still get around this.

File Uploads Rule #2

Always Validate The File Extension

Always validate file extension

- CF10 allows you to specify a file extension list in the accept attribute.
- You can also validate cffile.ServerFileExt
- Do both.

File Uploads Rule #3

Never upload directly to webroot

Don't upload to web root

- File can be executed before it's validated.
- Upload outside root, eg GetTempDirectory ram://, s3, etc.

Additional Tips

- Ensure upload directory can only serve static files.
- Consider keeping files outside webroot and serve with cfcontent or mod_xsendfile
- Specify mode on unix (eg 640 rw-r-----)

Path Traversal Vulnerabilities

Path Traversals

- Avoid file paths derived from user input.
- Strip and validate any variables used in paths.
- Beware of null bytes

Cross Site Scripting

XSS Vulnerable

```
<cfoutput>
  Hello #url.name#
</cfoutput>
```

hello.cfm?name=<script>...</script>

XSS

- XSS holes give attackers a CMS to create any content.
- Can be used to steal sessions
- Phish for passwords or other info.

Preventing XSS

- Strip out dangerous characters
  - < > ' " ( ) ; #
- Escape dangerous characters
  - CF10/Railo4 EncodeForHTML, etc.

Preventing XSS

Context	Method
HTML	`encodeForHTML(variable)`
HTML Attribute	`encodeForHTMLAttribute(variable)`
JavaScript	`encodeForJavaScript(variable)`
CSS	`encodeForCSS(variable)`
URL	`encodeForURL(variable)`

XSS in HTML

- Preventing XSS when allowing users to enter HTML is difficult.
  - AntiSamy
  - ScrubHTML

XSS Utils

- **Encoders**
  - ESAPI: http://www.petefreitag.com/item/788.cfm
  - OWASP Encoder: http://owasp-java-encoder.googlecode.com
- **Sanitizers**
  - AntiSamy: http://www.petefreitag.com/item/760.cfm
  - ScrubHTML: https://github.com/foundeo/cfml-security

Content-Security-Policy

- HTTP Response Header dictates what assets can be loaded. For example:
  - script-src 'self';
  - script-src 'self' cdn.example.com;
  - script-src 'none';
  - script-src 'unsafe-inline';

CSP Directives

- default-src
- script-src
- style-src
- img-src
- connect-src
- font-src
- object-src
- media-src
- frame-src
- sandbox
- report-uri

CSP Browser Support

- X-Webkit-CSP (Chrome 14+, Safari 6+)
- X-Content-Security-Policy (Firefox 4+, IE10)
    - IE10 support limited to sandbox directive.
- Unprefixed Content-Security-Policy Chrome 25+

CSP

- content-security-policy.com

Authentication

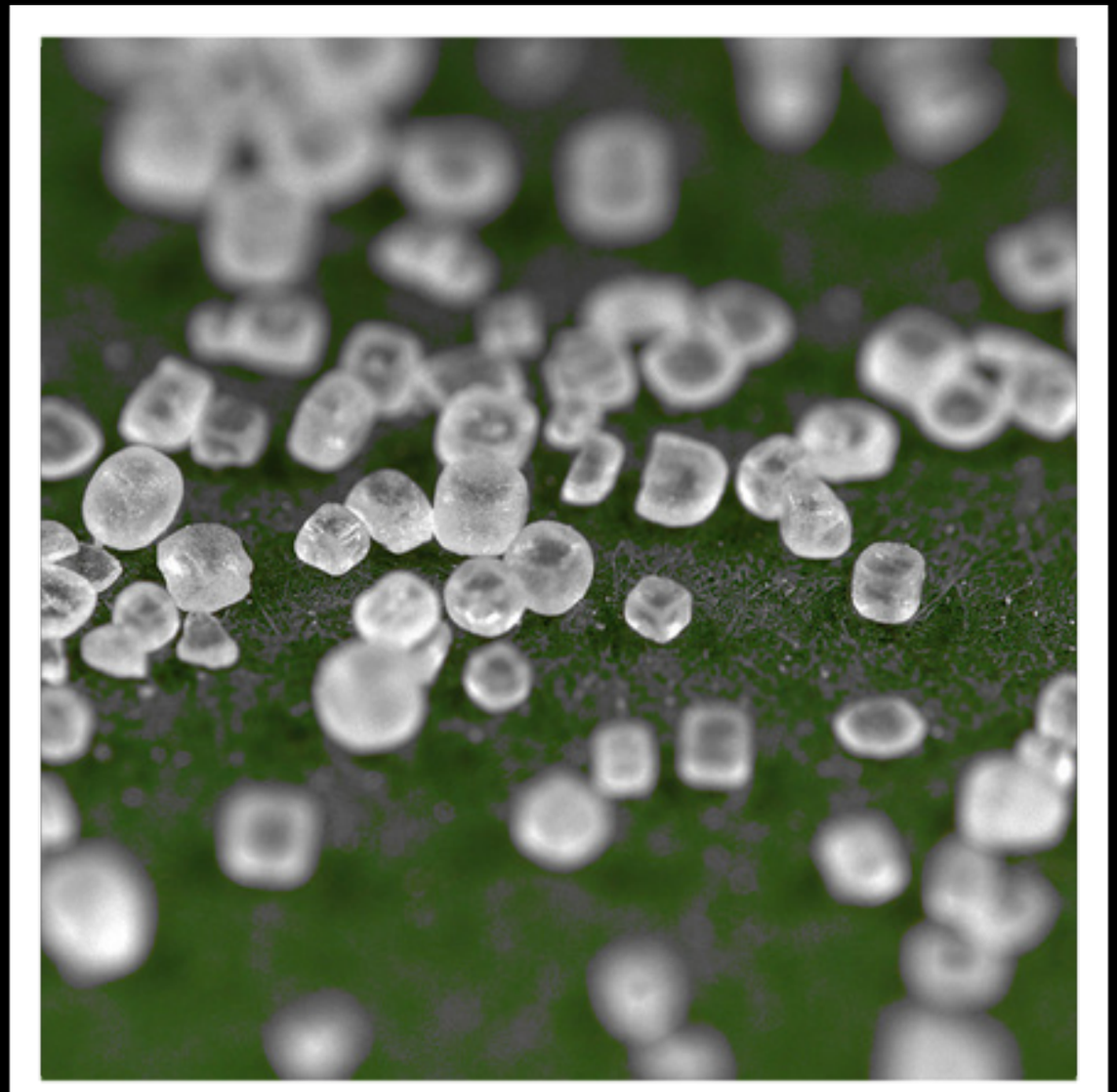
Lots of room for error

Authentication

- Store passwords hashed and salted
    - Builtin, Consider SHA-512
    - Don't use weak algorithms, eg MD5
    - Consider an adaptive one way function
        - bcrypt
        - scrypt
        - PBKDF2

Salt

- Cryptographically Random
- Unique for each credential
- Generate new when credential changes
- Sufficient length



Timing Attacks

```
<cfquery name="user">
    SELECT id, salt, password
    FROM user
    WHERE username = <cfqueryparam value="#form.username#">
</cfquery>
<cfif user.recordcount AND Hash(user.salt & form.password, "SHA-512") IS user.password>
    <cfreturn true>
<cfelse>
    <cfreturn false>
</cfif>
```

Authentication

- Prevent dictionary attacks
  - Ban IP or lock user after X attempts within Y seconds.
  - Require strong passwords
  - Consider limiting 1 login attempt per user per second, beware DOS
  - Requires keeping an audit log

Remember Me Cookies

- Not recommended for high security apps
- Don't base the token on the password
- Charles Miller Approach
  - Store random crypto token in DB with user id, timestamp, validity bit
  - Invalidate token when used and generate a new one.
  - Hash and Salt the token in the DB

Forgot Password

- Should be just as secure as your login code (rate limit, block after repeat fails)
- Use security questions but don't rely solely on them.

Session Fixation

Attacker generates a session on public computer

records the session id values, and leaves browser open

Session Fixation

Attacker generates a session on public computer

records the session id values, and leaves browser open

User logs into site

Session Fixation

Attacker generates a session on public computer

records the session id values, and leaves browser open

Attacker uses recorded session id to steal session

User logs into site

Session Fixation

- Phishing: Click here to update your account:
  - http://example.com/?cfid=1&cftoken=2

Preventing Session Fixation

- Call **SessionRotate** upon successful login
    - Does not rotate JSESSIONID, but does invalidate CSRF tokens.
- Call **SessionInvalidate** upon logout, or suspicious action
    - getPageContext().getSession().invalidate()

Session Hijacking

- Require SSL for authenticated users
- Use secure and httponly cookies
- Don't send session ids in url
    - Beware of cflocation

Session Cookie Settings

```
component {
   this.name = "sessionExample";
   this.sessionManagement = true;
   this.sessionTimeout = CreateTimeSpan(0,0,20,0);

   this.sessioncookie.httponly = true;
   this.sessioncookie.secure = true;
   this.sessioncookie.domain="example.com";
   this.sessioncookie.timeout=-1;
}
```

Authentication Resources

- http://stackoverflow.com/questions/549/the-definitive-guide-to-forms-based-website-authentication
- http://fishbowl.pastiche.org/2004/01/19/persistent_login_cookie_best_practice/
- https://www.owasp.org/index.php/Authentication_Cheat_Sheet

Encryption

- Protect the Private Key
  - Hardware Security Module
    - Offload encryption and key storage to hardware device
- Keys should be stored encrypted, and ideally separated from the web and DB server.
- Leverage java keystores, MS DPAPI

Encryption

- Don't use weak encryption algorithms or try to invent your own (eg CFMX_COMPAT)
- May need to install unlimited strength jurisdiction policy for stronger keys
  - GenerateSecretKey("AES", 256)
  - http://www.petefreitag.com/item/803.cfm

Last but not least

SQL Injection

SQL Injection

```
<cfquery name="news">
    SELECT id, title, story
    FROM news
    WHERE id = #url.id#
</cfquery>
```

news.cfm?id=1;delete+from+news

SQL Injection

- The solution - use cfqueryparam whenever possible.
- Validate and sanitize when you can't
  - ORDER BY *column*
  - SELECT TOP *10*
- ORM: make sure HQL statements are parameterized

Questions?

Thank You

foundeo.com | hackmycf.com | fuseguard.com