Securing your Box Applications

Brad Wood, Ortus Solutions
Pete Freitag, Foundeo Inc.

ortussolutions.com | foundeo.com

About Brad

- ColdBox Platform Evangelist
- CFML Programmer, DBA, Dad, Handyman
- blog: codersrevolution.com
- twitter: @bdw429s

About Pete

- 16 Years ColdFusion Experience
- 8 Years Foundeo Inc. Consulting & Products
    - IntoTheBox / cf.O Sponsor
- blog: petefreitag.com
- twitter: @pfreitag

Agenda

- File Upload Vulnerabilities
- Path Traversals
- Cross Site Scripting
- SQL Injection
- Password Security
- FuseGuard

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.

XSS DEMO

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
        - CF11: GetSafeHTML IsSafeHTML
    - 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
  - cfdocs.org/getsafehtml
  - cfdocs.org/issafehtml
  - ScrubHTML: https://github.com/foundeo/cfml-security

Content-Security-Policy

- HTTP Response Header dictates what assets can be loaded.
  - Come to Pete’s talk at cf.O tomorrow!

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 Demo

SQL Injection

- The solution - use cfqueryparam whenever possible.
- Validate and sanitize when you can't
    - ORDER BY *column*
    - SELECT TOP *10*

SQL Injection

- ORM: make sure HQL statements are parameterized. ORMExecuteQuery()
- CF11 / Railo4.2: queryExecute()
  - QueryExecute("SELECT x FROM y WHERE id = :id", {id=1});

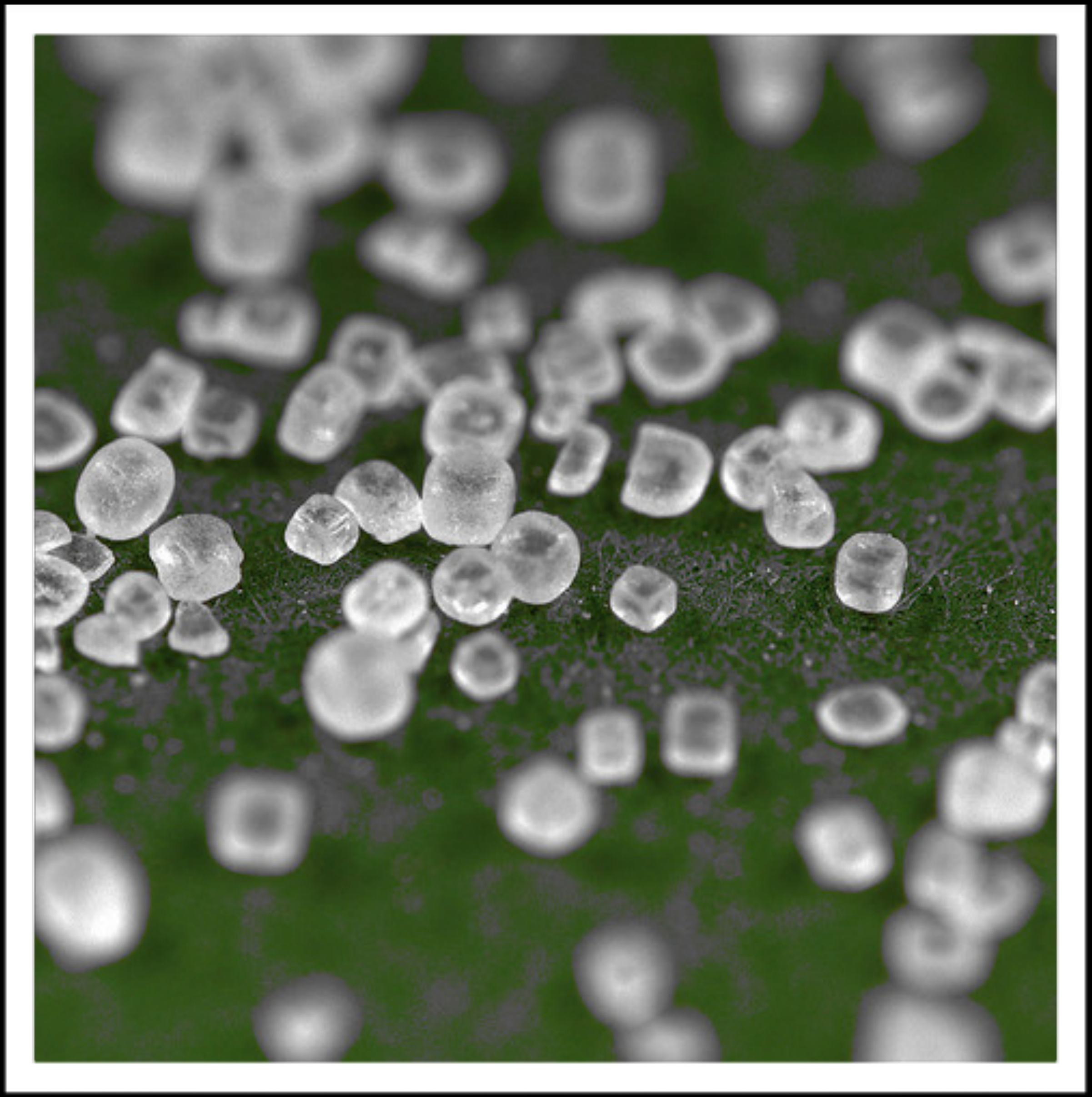
Password Security

Passwords

- Store passwords hashed and salted
  - Hash() builtin function
  - Don't use weak algorithms, eg MD5
  - Consider an adaptive one way function
    - bcrypt
    - scrypt
    - PBKDF2 - GeneratePBKDFKey()

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>
```

FuseGuard

What is FuseGuard?

- Web Application Firewall (WAF) for CFML written in CFML
- Logs / Blocks Malicious Requests
- Extensible & Configurable CFC API

Do you need a WAF?

- Can you write perfectly secure code?
- Do you understand all possible attack vectors?
- Have you reviewed ALL your source code for security vulnerabilities?

Do you need a WAF?

- Defense in Depth
  - Multiple layers of potentially redundant controls.
  - If one layer fails or is bypassed the secondary layer is there.
  - Example: A bank: locks front door, alarm system, security guard, vault, etc.

Do you need a WAF?

- Protect code you didn’t write (third party modules) or don’t understand (developer Spaghetti)
- Do you know when you are attacked?
- PCI Compliance:
  - Section 6.6 Requires that you either perform code reviews, or implement a WAF. Do Both!

WAF Weaknesses

- They require some configuration to work well.
- Often used as a crutch
- Can provide a false sense of security

Can I Ditch Secure Coding Practices and Use a WAF?

Absolutely not

FuseGuard

DEMO

FuseGuard Pricing

- Ortus FuseGuard ColdBox Module
    - Application License: $449
    - Server License: $1199
    - Enterprise: $8999

Questions?

Thank You

ortussolutions.com | foundeo.com