Hack proofing ColdFusion

Shlomy Gantz
FAQ

Yes, it’s 2 hours long.

Yes, slides will be available after at

http://www.shlomygantz.com
<CFSET CurrentTitle =“President, BlueBrick Inc.”>  

<CFSET experience_YY = 16>  
<CFSET experience_CF = 12>  
<CFSET experience_PM = 12>  

<CFSET aTitles = arrayNew(1)>  
<CFSET aTitles[1] = “Adobe Certified Instructor”>  
<CFSET aTitles[3] = “Manager, NYFLEX user group”>  
<CFSET aTitles[4] = “Speaker, CFUNITED, Max..”>  

<CFSET Mom = “Very Proud”>
Agenda

- ColdFusion Vulnerabilities (OWASP)
  - Concepts
  - Real Code
  - Demonstration

- Beyond OWASP
  - Admin
  - RIA
  - SDLC

- Q&A
Layers Of Vulnerability

Social Engineering

Application

Web Server

Network / OS

Oooohhh .. Fancy animation… … get to the code already
Application Attacks

• Relate To The Meaning Of Application Messages:
  ▪ Interpretation of The HTTP Requests
  ▪ Handling of SQL Queries
  ▪ Interpretation of Application Specific Messages

• Harder To Identify Or Replicate
  ▪ Requires understanding of both technology and application domain
  ▪ Vulnerabilities differ between applications
Application Attacks

- Easier To Exploit...
  - Coding Is Simple
  - GUI Assisted (Paros, NetCat…)

I know “Code Fu”
Application Attacks

- Application Attacks Are Often More Dangerous
  - Involve Organization’s Core Operation
  - Infrastructure Attacks Usually Target The Servers Themselves Only
- Harder To Repair…
  - May Require Code and Design Changes
  - Most Security Staff Has IT Background Rather Than Development Background
The origin of vulnerabilities

- Applications assume certain client behaviors
- Developers anticipate only “Real” users will input data

ALL Input Can Be Modified
OWASP top 10 list

1. Unvalidated Input
2. Broken Access Control
3. Broken Account and Session Management
4. Cross-site Scripting (xss) Flaws
5. Buffer Overflows
6. Command Injection Flaws
7. Error Handling Problems
8. Insecure Storage
9. Application DoS
10. Insecure Configuration

That is so 2004…
OWASP TOP 10 - 2007

• A1 - Cross Site Scripting (XSS)
• A2 - Injection Flaws
• A3 - Malicious File Execution
• A4 - Insecure Direct Object Reference
• A5 - Cross Site Request Forgery (CSRF)
A6 - Information Leakage and Improper Error Handling
A7 - Broken Authentication and Session Management
A8 - Insecure Cryptographic Storage
A9 - Insecure Communications
A10 - Failure to Restrict URL Access
A1 - XSS

- Execute scripts in the victim’s browser
  - Hijack user sessions/info
  - Deface web sites
  - Insert hostile content
  - Phishing attacks

```javascript
<script language="javascript">
</script>
```

EXAMPLE 1
A1 - XSS

- **Stored**
  - Script Is Stored in Trusted Source
    - a) Forums
    - b) User Comments
    - c) Contact Forms
    - d) Online Web Mail System

- **Reflected**
  - Script Reflected Off The Web Server In
    - Error Messages
    - Search Results
A1 – XSS – Mitigation

- **Input validation**
- **Was OWASP top ten # 1 in 2004**
  - The Most Simple Form Of Application Attack
  - Targets The Business Logic Of The Application
  - Does Not Require Any Special Tools
  - Can Be Done On Both Get and Post Variables
A1 – XSS – Mitigation - Input

- **Forms**
  - Input fields
  - Hidden Fields
  - User Selection

- **URL**
  - Query String parameters
CartStep1.cfm
<input type="hidden" name="price" value="250">

Register.cfm
<select name="role">
    <option>User</option>
    <option>Client</option>
</select>

<!-- <option>Admin</option> -->
A1 – XSS – Mitigation - Input

- Reduce Dependency On Hidden Fields By Using The Session Scope
- Do Not Rely On Client Side Validation Alone
- Check Validity Of User Selection and Input Type/Range
  - Use `<CFPARAM>` `Type`, `Pattern` and `Range` attributes (min, max)
  - Use `<CFINPUT>` `validate` attribute onServer as well as onBlur
A1 – XSS – Mitigation - Input

- Check validity of data using
  - isValid()
  - isDate(), isNumeric(), isMonkey()

- Enforce maximum length
  - left()
A1 – XSS Mitigation

- **Strong output encoding**
  - Built in function
    - HTMLEditFormat()
  - HtmlTrans()
    - [http://www.cflib.org/udf.cfm?id=945](http://www.cflib.org/udf.cfm?id=945)

- **Specify encoding** (such as ISO 8859-1 or UTF 8). Do not allow the attacker to choose this for your users.
A1 – XSS - Mitigation

• Do not use "blacklist" validation, but…
  ▪ Built in CF Protection ScriptProtect
    ▪ CF Admin Setting or in Application.cfc
      ▪ None
      ▪ All - Form, URL, CGI, and Cookie)
      ▪ List of ColdFusion scopes
  ▪ Uses RegEx in neo-security.xml to remove:
    ▪ <object>
    ▪ < embed >
    ▪ < script >
    ▪ < applet >
    ▪ < meta >
Do not use "blacklist" validation, but…

- CF_XSSBLO CK
- Change neo-security.xml

- Log, Alert and Review Violations!
A2 – Injection Flaws

- executing unintended commands or changing data.
  - SQL Injection
  - HTML Injection (Huh?)

EXAMPLE 2 – SQL Injection
EXAMPLE 3 – HTML Injection
A2 – Injection Flaws

- SQL Injection

```cfquery name="qUser">
SELECT * FROM users WHERE user_id = #url.user_id#
</cfquery>

editUser.cfm?user_id=1;delete from users
A2 – Injection Flaws

• SQL Injection can be used for
  ▪ Executing malicious code
  ▪ Circumventing Security
  ▪ Stealing information
  ▪ Defacing sites (Cast)

    DECLARE@S CHAR(4000); SET@S=CAST(0x4445434C45 2452040542076E61726368617228323535

   ▪ Adds text to all text/char fields

getProduct.cfm

<cfset strSQL = "SELECT * FROM tblUser WHERE user_id ="">
<cfset strSQL = strSQL & url.user_id>

<cfquery name="qUser">
#PreserveSingleQuotes(strSQL)#
</cfquery>
A2 – Injection Flaws - Mitigation

- `<CFQUERYPARAM>`
- Consider Stored Procedures
- Limit DB Permissions
  - CF Admin datasource settings
  - Database
- Disable XP_cmdshell and Equivalents
- Consider Server Sandboxing
- Do not rely on ColdFusion s. quote escaping
A2 – Injection Flaws - Mitigation

- **Tools:**
  - Free: wsdigger, sqlmap

- **Cheat Sheet**

A3 - Malicious File Execution

- Remote file inclusion (RFI)
  - any framework which accepts filenames or files from users.
- Remote code execution
- Remote root kit installation
addClientLogo.cfm
<cffile action="upload"
    destination="#expandpath('.', )#\images\logos"
    filefield="theFile" nameconflict="makeunique">

logo: c:\hack\hglogo.gif.cfm
A3 - Malicious File Execution – The result

All your base are belong to us…

Zero wing
<cfif IsDefined("FORM.cmd")>
  <cfoutput>#cmd#</cfoutput>
  <cfexecute name="C:\windows\System32\cmd.exe"
    arguments="/c #cmd#"
    outputfile="#GetTempDirectory()#foobar.txt"
    timeout="1">
  </cfexecute>
</cfif>

...
A3 - Malicious File Execution - Mitigation

- Securing `<CFFILE action="upload">`
  - Use ACCEPT attribute
    `<cffile action="upload" accept="image/gif,image/jpeg" ...>`
  - Do not rely on MIME type alone, confirm on server side
  - Use Built in ColdFusion function
    a) `isImageFile()`
    b) `isPdfFile()`
    c) `isMonkeyFile()`
A3 - Malicious File Execution - Mitigation

- Upload files outside of webroot
  - Serve them back with CFCONTENT
- Limit file size by looking at cgi.content_length
  - CGI.CONTENT_LENGTH/1000 ~ KB
- Consider renaming files, use indirect /stored references
- Change permissions on uploaded files
A4 - Insecure Direct Object Reference

- Exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter
A4 - Insecure Direct Object Reference

- For example Primary Keys

```cfc
viewAccount.cfm?accountNo=455324143
<cfquery name="bankdb">
SELECT * FROM tblAccounts
WHERE AccountNo = '#url.accountNo#'
</cfquery>
```
### Primary Keys

- **Switch to UUID**
  - `CreateUUID()`
- **Consider using internal/external references**

<table>
<thead>
<tr>
<th>UserID</th>
<th>SUserID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>550e84000e29b41d4a7164466554400000</td>
<td>1</td>
<td>Shlomy</td>
</tr>
<tr>
<td>450e8400edf87ac145466111343474362</td>
<td>2</td>
<td>Jane</td>
</tr>
</tbody>
</table>
Use Hash() to perform checksum

- One way transformation
- Almost impossible to reverse

`?user_id=#user_id#&chk=#hash(user_id)#`

- Checksum is performed on the next page

```cfc
<cfif hash(user_id) is not chk>
  <cfabort>
</cfif>
```
A5 - Cross Site Request Forgery (CSRF)

- Rely on victim to be logged in
- Request is sent using victim’s browser while they are logged in.
  - Perform administrative functions
  - Perform bank transactions
A5 - Cross Site Request Forgery (CSRF)

1. Admin Logs in to discussion forum

2. Admin views messages in forum

3. Message contains
   - `<img src="deleteUser.cfm?user_id=1">`

4. User 1 is deleted
A5 - Cross Site Request Forgery (CSRF)

- Use POST instead of GET
  - and scope your variables !!!

- Ask for re-authentication on sensitive pages
  - Amazon does…

- Insert custom random tokens into every FORM and URL requests
A5 - Cross Site Request Forgery (CSRF)

list.cfm
<cfset session.csrf.userDelete = CreateUUID() />
<a href="deleteUser.cfm?user_id=#qUser.User_ID#&chk=#session.csrf.userDelete#">
<cfif NOT url.chk is session.csrf.userDelete>
<CFABORT>
</cfif>
<cfset structDelete(session.csrf,"userDelete")>
Applications can unintentionally leak information about
- Configuration
- Internal workings
- Sensitive data

Detailed error handling
Detailed exception/validation messages
A6 - Information Leakage and Improper Error Handling

• Detailed Error can disclose:
  - Directory Structure
  - Code Snippets
  - Query Structure

• Detailed exception/validation can disclose:
  - A better vector of attack
  - Private information
A6 - Information Leakage and Improper Error Handling

- Disable Debugging on Production
- Define Site Wide Error and 404 Handler
  - BTW: Review error AND 404 logs
- Use `<CFERROR>` / `OnError()`
  - You can disable them based on a session variable
- Display error codes for sensitive validation/exceptions
- Only display “login failed” on failed authentication
A7 - Broken Authentication and Session Management

- Account Authentication Bypassing
  - Login Tampering
  - Brute Force

- Session Hijacking
  - Brute Force
  - ID Predicting
  - Sniffing and Eavesdropping
  - Using HTTP_REFERER with SessionID Is Passed On URL
A7 - Broken Authentication and Session Management

- Enforce At Least 8 Characters Password
- Require Numbers and Special Characters
- Do Not Send Permanent Passwords Via Email
- Expire Passwords
- Do not allow repeat passwords
- Restrict Access After Failed Login Attempts
- Log, Alert and Review!
A7 - Broken Authentication and Session Management

- Require Re-authentication On Email Change
- Use SSL on login page
- Regenerate session on authentication
- Disable Browser Caching
  - Prevents cached data from being accessed
- Use UUID For CFTOKEN
- Use J2EE Sessions
- Control Session Timeout
A7 - Broken Authentication and Session Management

- Check CGI Variables
  - CGI. HTTP_REFERER
  - CGI. CF_TEMPLATE_PATH
  - Note: They Can Be Spoofed!
- `<CFLOGIN>` functions
  - IsUserInRole(), getAuthUser(), IsUserInAnyRole()
  - GetUserRoles(), IsUserLoggedIn()
- `<CFNTAuthenticate>`
A7 - Broken Authentication and Session Management

• Set Session Cookies to HTTPOnly
  - Jason Dean
Storing Sensitive Information Using Inadequate Encryption Schemas

- Failure to encrypt critical data
- Insecure storage of keys, certificates, and passwords
- Improper storage of secrets in memory
- Poor sources of randomness
- Poor choice of algorithm
- Attempting to invent a new encryption algorithm
- Failure to include support for encryption key changes and other required maintenance procedures
Oh… and one more thing…

Storing passwords in clear text

<table>
<thead>
<tr>
<th>Password</th>
<th>Username</th>
<th>Password 1</th>
<th>Password 2</th>
<th>Password 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>“password”</td>
<td>“test”</td>
<td>“admin”</td>
<td>“demo”</td>
<td></td>
</tr>
<tr>
<td>“pass”</td>
<td>“test”</td>
<td>“1234556”</td>
<td>“jesus”</td>
<td></td>
</tr>
<tr>
<td>“sunshine”</td>
<td>“princess”</td>
<td>“love”</td>
<td>“iloveyou”</td>
<td></td>
</tr>
</tbody>
</table>
A8 - Insecure Cryptographic Storage

- Encrypt Sensitive Data
  - Encrypt()/Decrypt() – Two Way
    a) Uses Symmetric Key
    b) CF7
      - Additional Algorithms (AES, BLOWFISH, DES…)
      - Generatesecretkey()
    c) CF8
      - RSA BSafe encryption
  - Hash() – One Way
    a) Nearly Impossible To Revert
    b) Does Not Require Key
    c) Best For Passwords
    d) Adding “Salt”
A9 - Insecure Communications

- Failure to encrypt sensitive communications
- Use SSL for transmitting sensitive or value data
  - Credentials
  - Credit card details
  - Health
  - Private information
- PCI DSS compliance is mandatory for merchants and anyone else dealing with credit cards.
A10 - Failure to Restrict URL Access

- Only preventing the display of links or URLs to unauthorized users.
  - Accessing unauthorized action
  - Accessing unauthorized files
A10 - Failure to Restrict URL Access – Code

ListUser.cfm

<tr>
  <td>#name#</td>
  <td>
    <cfif session.role is "admin"> 
    <a href="edit.cfm?user_id=#user_id#">edit</a>
    </cfif>
  </td>
</tr>
A10 - Failure to Restrict URL Access – Code

showContractPDF.cfm

<cfinclude template="#url.ID#/url.ID#.pdf"
A10 - Failure to Restrict URL Access

- Check Data Access Permissions On Every Request
- Control Access from a single location
  - Rely on session level variables rather than cookies
A10 - Failure to Restrict URL Access

- Files
  - Path Traversal
    - Getfile.cfm?file=head1.pdf
  - While we are on the subject …
    a) Forceful Browsing
      - Static File Links
      - Hidden Files (Security By Obscurity)
      - File Name Predictions
        i. Known System Files
        ii. .log / .old Files
        iii. Structured file name
A10 - Failure to Restrict URL Access

- Files
  - Store Files to Download Outside Of Webroot
  - Use `<CFCONTENT>` To Serve Files To The User

```
<CFCONTENT TYPE="application.pdf"
  FILE="d:\contracts\#id#.pdf">
```

- Block access to .xml or .ini and similar files at the web server level
Beyond OWASP – CF Administrator

• Configure CF Admin
  ▪ New in CF8
    a) RDS sandbox support
    b) User-based Administrator access
  ▪ Secure Admin Directory With NT Authentication Or Completely Remove
  ▪ Do Not Deploy Docs, Sample Apps and RDS To Production
  ▪ Do Not Store DB Password in code
  ▪ Disable Unused Services
Beyond OWASP – CF Administrator

• If you are not using, disable
  ▪ “Flash remoting”
  ▪ “access to internal ColdFusion Java components”
  ▪ “Watch configuration files for changes”
• Set default request timeout ~10 seconds
  ▪ You can override for long running requests
• Check “Prefix serialized JSON with” (//)
Beyond OWASP – CF Administrator

- Enable global Script protection
- Set Maximum size of post data
- Change client variable storage from registry
- Use operating system logging facilities
Beyond OWASP – CF Administrator

- **Set Privileges**
  - Run ColdFusion service as a user
  - User name and password authentication for CFADMIN
  - Create least privilege user for each DSN

- **Sandbox applications**
  - Remove execute from non cfm folders

Beyond OWASP - RIA Security

- CFCs can be used as back-end for:
  - Flash/Flex
  - AJAX
  - SOAP
  - Non-browser based application

- New in CF8
  - VerifyClient()
  - secureJSON()
Beyond OWASP - SDLC

- Integrate Security Into Your SDLC
  - Design with security in mind
  - Hack/Pen Test During/After Development
  - Create Anti-requirements
  - Review Code Regularly

- Hack proofing old code
  - Automate the process
  - Follow a checklist based on OWASP
Beyond OWASP - SDLC

- Security Analysis
  - Define threats
    a) data
    b) Architecture
  - Assess the Impact (cost/benefit)
    a) Financial
    b) PR
  - Mitigate
Beyond OWASP - SDLC

• Define threats
  ▪ STRIDE
    1. Spoofing Identity
    2. Tampering with Data
    3. Repudiation
    4. Information Disclosure
    5. Denial of Service
    6. Elevation of Privilege
Beyond OWASP - SDLC

• Assess the Impact
  ▪ DREAD (score 1-10 .. Then avg.)
    1. Damage Potential – Houston, we have a problem
    2. Reproducibility – Happens every time
    3. Exploitability – A monkey could do it
    4. Affected Users – TJX 45.7 million cards
    5. Discoverability – No source code required

• Mitigate
Beyond OWASP – Security Design Principles

- Authentication
  - Who are you
- Authorization
  - What can you do
- Confidentiality
  - What can you see
- Non-repudiation
  - Did you really do that
- Availability
  - Your ability to do it (no nike pun please)
Beyond OWASP - Social Engineering

• Simply Asking For:
  ▪ Information
  ▪ Passwords
  ▪ Assistance

• Requires No Technical Skills
Resources

- Open Web Application Security Project (OWASP)
  - http://www.owasp.org

- ColdFusion Specific
  - http://www.coldfusionsecurity.org/

- Products
  - Foundeo Web Application Firewall for ColdFusion
Questions

Shlomy Gantz

shlomy@bluebrick.com

http://www.shlomygantz.com/blog