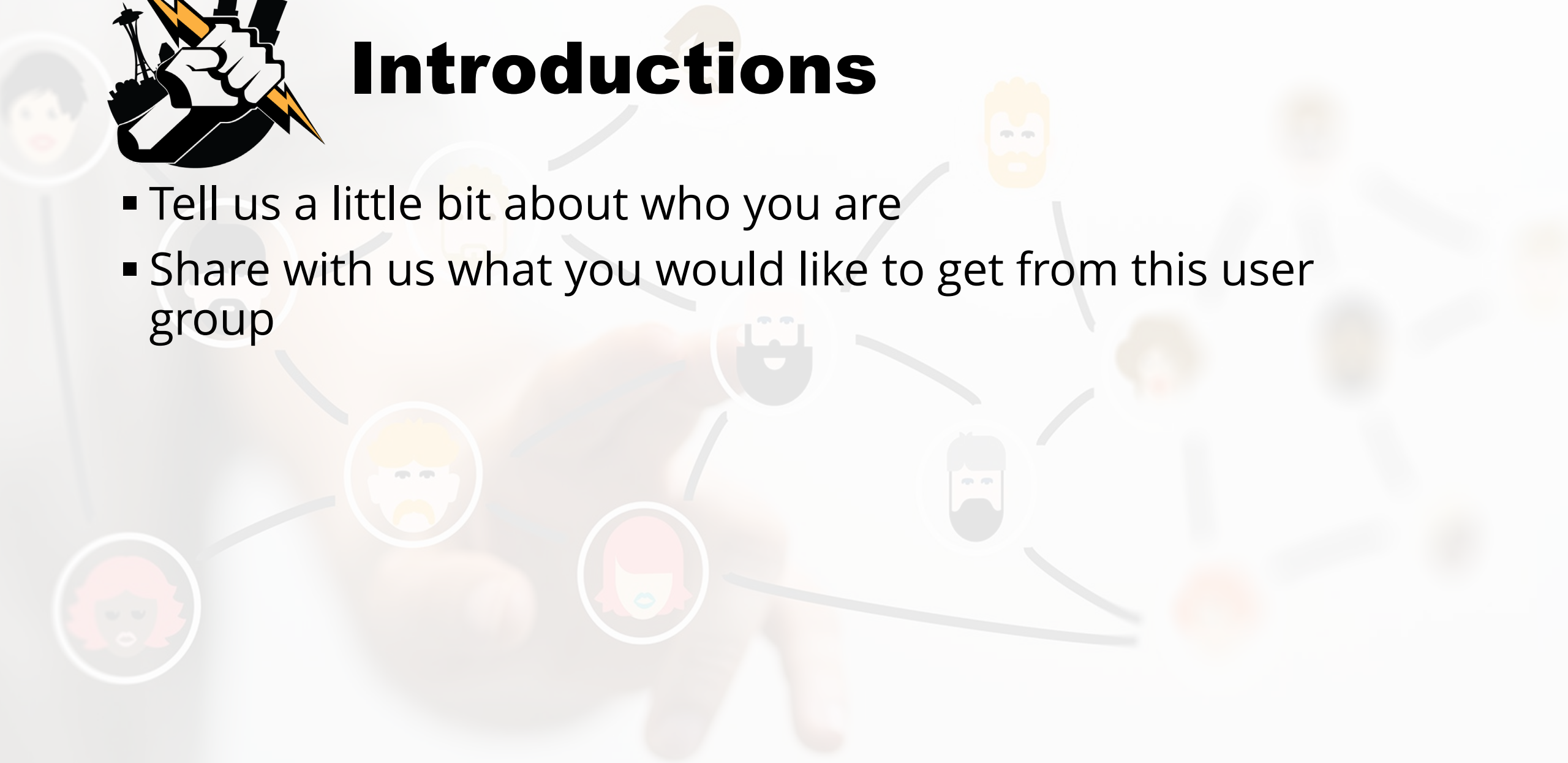

November
2018
ColdFusion
user group Seattle

Agenda

- Welcome to the Seattle ColdFusion User Group
- Introductions
- Goals
- How to protect your applications from SQL Injection Attacks
- How to protect your applications from XSS Attacks
- How to protect your web application from click-jacking
- CF Alive
- Next Steps for the Seattle ColdFusion User Group
- December 2018 Meeting
- Questions/Answers/Help Needed

Introductions

- Tell us a little bit about who you are
- Share with us what you would like to get from this user group



Goals

- Assist ColdFusion Developers Throughout the Pacific Northwest
- Promote ColdFusion Developers Throughout the Pacific Northwest
- Connect Employers with ColdFusion Developers
- Establish a Community of Friendship Between ColdFusion Developers
- Provide Speaking Opportunities for ColdFusion Developers
- Change the Perception of ColdFusion as a viable platform

How to protect your web application from SQL Injection Attacks

What is a SQL Injection (SQLi) Attack?

- **SQL injection** is a code injection technique, used to attack data-driven applications, in which nefarious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).
- SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed.

How to protect your web application from SQL Injection Attacks

How can I protect my application from this type of attack?

- Utilize the Principle of Least Privilege
- Use the appropriate data type for each field in you database
- Validate all user provided data PRIOR to it touching the database
- USE CFQueryParam and CFProcParam tags with all of your queries/stored procedures

How to protect your web application from SQL Injection Attacks

Principle of Least Priviledge

Each query or stored procedure call should utilize the lowest permission level possible required to execute that query or stored procedure

How to protect your web application from SQL Injection Attacks

Use the appropriate datatype for each database field

- Avoid using varchar/text other field that accept free-form characters when data would not be free-form
  - Dates/Times
  - Numbers

How to protect your web application from SQL Injection Attacks

Validate all user provided data PRIOR to it touching the dB

- Validate all user provided data client-side that it conforms to what is expected
- Use the following functions to validate data server side: IsNumeric, IsDate, IsValid

How to protect your web application from SQL Injection Attacks

Use CFQueryParam and CFProcParam tags with each query/stored procedure parameter

- Validate all user provided data client-side that it conforms to what is expected
- Use the following functions to validate data server side: IsNumeric, IsDate, IsValid

How to protect your web application from SQL Injection Attacks

CFQueryParam/CFQueryParam SQL Types

CF_SQL_VARCHAR, CF_SQL_INTEGER, CF_SQL_DATE, CF_SQL_SMALLINT, CF_SQL_NUMERIC, CF_SQL_TINYINT

- many more listed at https://cfdocs.org/cfqueryparam
- If using CF_SQL_NUMERIC don't forget to identify the scale (number of numbers to the right of the decimal point)
- No longer need to include CF_SQL_ for CF11+/Lucee 4.5+

How to protect your web application from SQL Injection Attacks

Additional Tips to prevent SQL Injection from free-form text

- Set the maxlength attribute in the CFQueryParam/CFProcParam tag
- Use the isSafeHTML function (OWASP AntiSamy) to check input provided from a rich text editor/textarea field

How to protect your web application from SQL Injection Attacks

Resources

- CFQUERYPARAM - https://cfdocs.org/cfqueryparam
- CFPROCPARAM - https://cfdocs.org/cfprocparam
- isSafeHTML - https://cfdocs.org/issafehtml
- isValid - https://cfdocs.org/isValid
- isNumeric - https://cfdocs.org/isNumeric
- isDate - https://cfdocs.org/isDate

How to protect your web application from XSS Attacks

What is XSS (Cross-Site Scripting)?

Cross-site scripting (**XSS**) is a type of computer security vulnerability typically found in web applications. **XSS** enables attackers to inject client-side scripts into web pages viewed by other users. A **cross-site scripting** vulnerability may be used by attackers to bypass access controls such as the same-origin policy.

How to protect your web application from XSS Attacks

What is XSS (Cross-Site Scripting) - continued?

#1 Vulnerability in Web Applications

Types of Cross-Site Scripting (XSS)

- Refected
- Persistant
- DOM

How to protect your web application from XSS Attacks

Reflected Cross-Site Scripting (XSS)

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

index.cfm?name=<script>alert(‘gotcha’)</script>

How to protect your web application from XSS Attacks

Reflected Cross-Site Scripting (XSS) – how to defense

1. Ensure debugging is turned off
2. Use the scriptprotect attribute the CFApplication tag or this.scriptprotct in an Application.cfc file (an okay 1st line of defense, but not a catch-all)

How to protect your web application from XSS Attacks

Reflected Cross-Site Scripting (XSS) – how to defense

3. Use the ESAPI functions around all output variables (unless the output variable is from a texarea field – then consider using the getSafeHTML function)
   EncodeForHTML, EncodeForHTMLAttribute, EncodeForJavaScript, EncodeForURL,EncodeForXML
4. Use Foundeo's CFML Security Utilities - https://github.com/foundeo/cfml-security/tree/master/securityutil

How to protect your web application from XSS Attacks

Reflected Cross-Site Scripting (XSS) – how to defense

5. Use the **X-XSS-Protection** response header
   not supported by Firefox

   <cfheader name="X-XSS-Protection" value="1;
   mode=block">
   or
   X-XSS-Protection: 1; mode=block as a custom header in
   web.config

How to protect your web application from XSS Attacks

Reflected Cross-Site Scripting (XSS) – how to defense

5. Use the **X-XSS-Protection** response header
   not supported by Firefox

   more info: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

How to protect your web application from XSS Attacks

Persistant Cross-Site Scripting (XSS)

The persistent (or stored) XSS vulnerability is a more devastating variant of a cross-site scripting flaw: it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping.

How to protect your web application from clickjacking

What is Clickjacking?

Clickjacking is an exploit that fools a web site user to interact with a site for the purposes of the attacker.

Typically, the attacker will:

- Have done extensive research on the particular web site to exploit
- Determine which functionality on the site to exploit (for example, a person's bank account), and include that content as an iFrame on their web site
- Take advantage of the logged-in user to have them perform actions on the web site to their benefit (examples of exploits involved transferring money to the attacker's account, to pad Facebook Likes for an individual (among many others)

How to protect your web application from clickjacking

How Do I Protect My App Against This?

Do not allow your web site to be included within an iFrame by an attacker.

How to protect your web application from clickjacking

What Are the Protections I can Add?

There are several things that you should include to provide the widest protection for users of older to the most modern web browsers

1. **Add the following CFHEADER tags to your Application.cfm or Application.cfc**

   <cfheader name="Content-Security-Policy" value="frame-ancestors 'none'">
   <cfheader name="X-Content-Security-Policy" value="frame-ancestors 'none'">
   <cfheader name=" X-WebKit-CSP" value="frame-ancestors 'none'">

   - or –

   <cfheader name="Content-Security-Policy" value="frame-ancestors 'self'">
   <cfheader name="X-Content-Security-Policy" value="frame-ancestors 'none'">
   <cfheader name=" X-WebKit-CSP" value="frame-ancestors 'none'">

   more info: https://caniuse.com/#search=Content%20Security%20Policy%201.0

How to protect your web application from clickjacking

What Are the Protections I can Add?

2. **Add the following CFHEADER tag to your Application.cfm or Application.cfc**

   <cfheader name="X-Frame-Options" value="DENY">

   - or -

   <cfheader name="X-Frame-Options" value="SAMEORIGIN">

   This option fills the gap for many other browsers (but not all) that do not support the **content-security-policy** header

   see: https://caniuse.com/#search=X-Frame-Options%20HTTP%20header

How to protect your web application from clickjacking

What Are the Protections I can Add?

3. **Add a “Best-for-now Legacy Browser Frame Breaking Script”**
   In the document HEAD element, add the following:

```
<style id="antiClickjack">body{display:none !important;}</style>
```

And then delete that style by its ID immediately after in the script:

```
<script type="text/javascript">
    if (self === top) {
        var antiClickjack = document.getElementById("antiClickjack");
        antiClickjack.parentNode.removeChild(antiClickjack);
    } else {

        top.location = self.location;

    }
</script>
```

How to protect your web application from clickjacking

What Are the Protections I can Add?

4. If you would prefer, and you are using IIS, add the following code to your web.config file under <configuration><system.webServer><httpProtocol><customHeaders>

```
<clear />
<add name="Content-Security-Policy" value="frame-ancestors 'self' X-Frame-Options: SAMEORIGIN;upgrade-insecure-requests" />
<add name="X-Content-Security-Policy" value="frame-ancestors 'none'" />
<add name="X-WebKit-CSP" value="frame-ancestors 'none'" />
```

Download example web.config

How to protect your web application from clickjacking

Example

Clickjacking Allowed:
https://www.seattlecfug.org/presentations/clickjacking.cfm

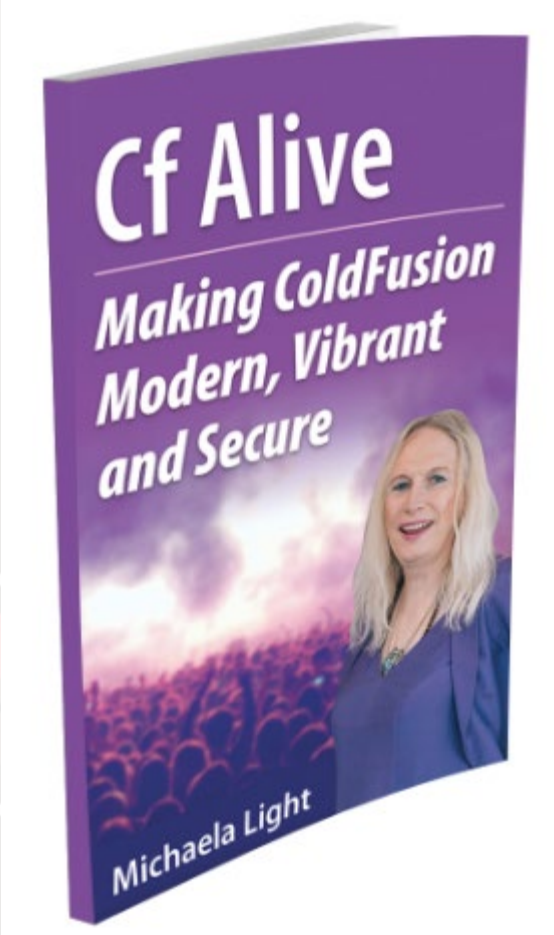
Clickjacking Not Allowed:
https://www.seattlecfug.org/presentations/clickjackingDisallowed.cfm

How to protect your web application from clickjacking

References

1. OWASP Clickjacking Defense Cheat Sheet: https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet
2. Can I Use? https://caniuse.com
3. Blackhat 2013 – Clickjacking Revisited - A Perceptual View of UI Security: https://www.youtube.com/watch?v=KUoHW3Eq-n4
4. Burp Suite Professional (will allow you to navigate to a page and write the script to test if a site is vulnerable to clickjacking) - https://portswigger.net

CF Alive



- Book Recently Released by Michaela Light

Next Steps for the Seattle ColdFusion User Group

- Does this venue work for you?
- Do you prefer meeting in person or online?
- What would you like our group to focus on?
- What topics would you like to hear?

Next Month's Meeting

- December 5, 2018 – WeWork – Lincoln Square – Bellevue Conference Room 5K

Questions/Answers/Help!