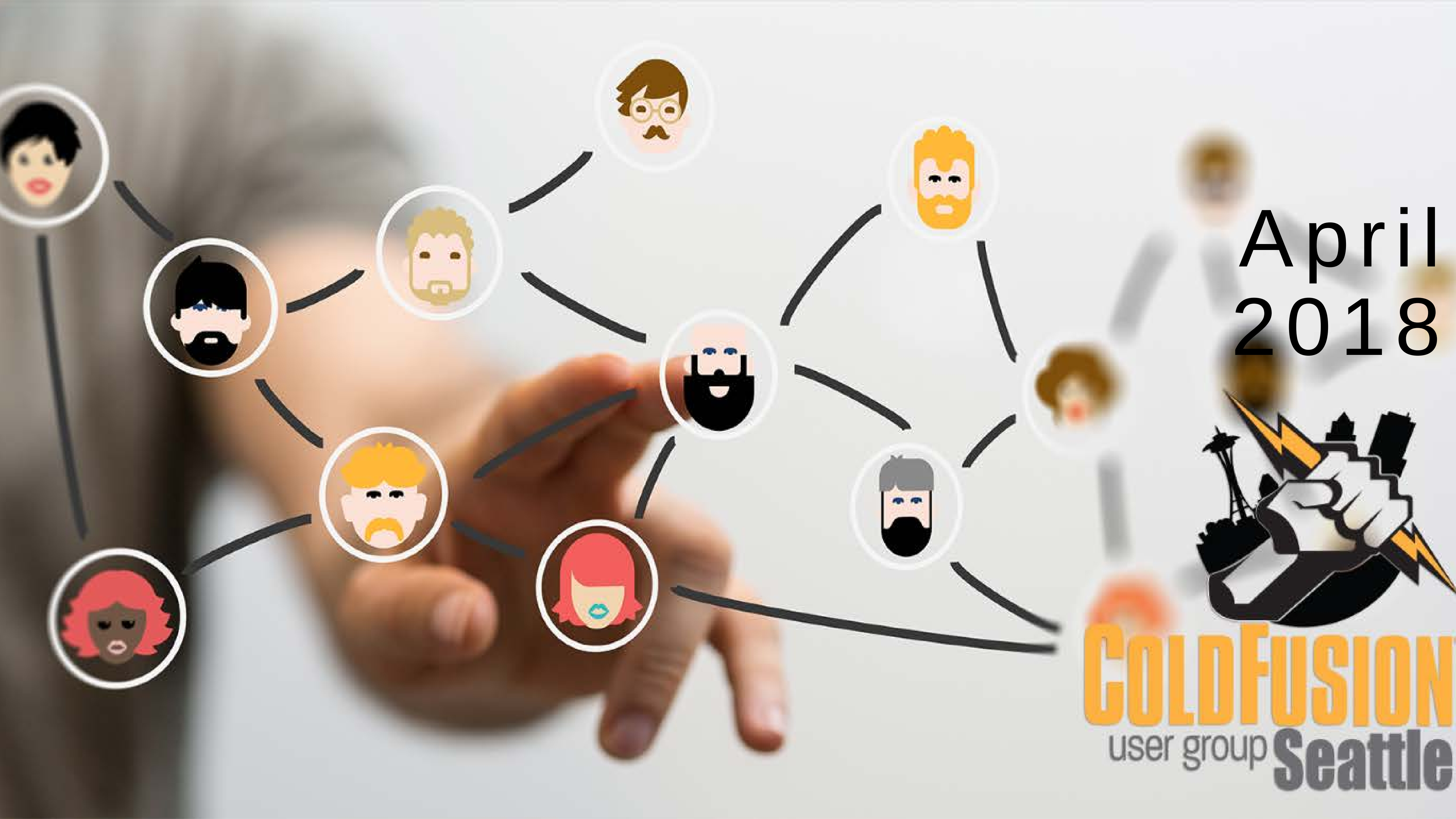


April
2018

ColdFusion
user group Seattle

Agenda

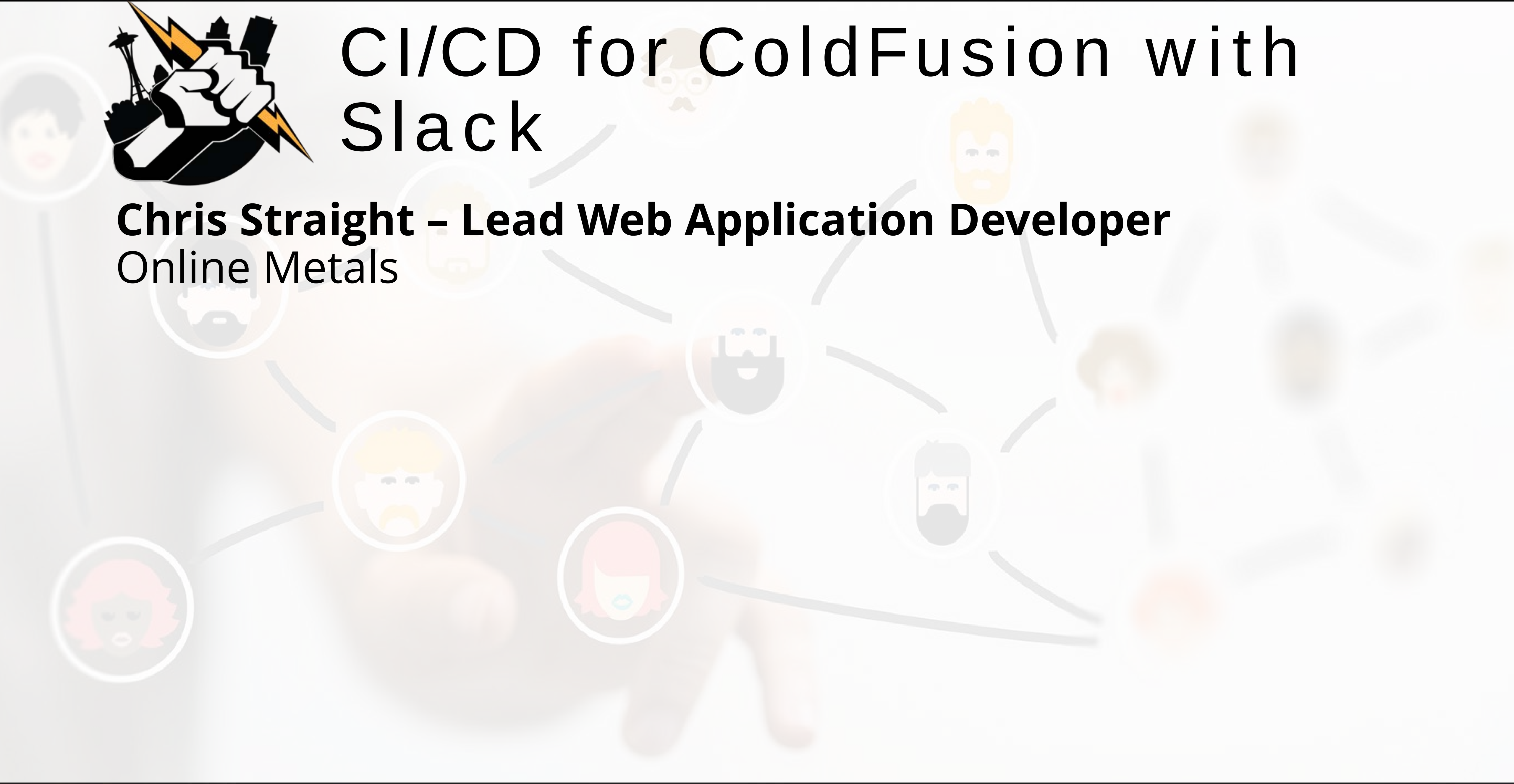
- Welcome to the Seattle ColdFusion User Group
- Introductions
- Goals
- CI/CD for ColdFusion with Slack
- How to protect your web application from click-jacking
- Why SALTing your passwords is a GOOD thing
- Upcoming ColdFusion Events
- Next Steps for the Seattle ColdFusion User Group

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



CI/CD for ColdFusion with Slack

Chris Straight – Lead Web Application Developer
Online Metals

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/

Why SALTing Your Passwords is a Good Thing

- Storing passwords in your database is NOT an ideal thing
- Storing password unencrypted is an even worse thing
- Passwords should be hashed using the SHA-512 algorithm to provide a good level of assurance that a user’s password cannot easily be compromised. This is accomplished using a **SALT**

Why SALTing Your Passwords is a Good Thing

What is a SALT (secret)…and why should I use one?

- A SALT (in this context) is a string that is used to help encrypt/decrypt a password
- The SALT would be created using the following function: <cfset variables.salt = Hash(GenerateSecretKey("AES"), "SHA-512") />
- The password would be hashed using the SALT as follows: <cfif variables.hashedPassword = Hash(form.password & variables.salt, "SHA-512") />

Why SALTing Your Passwords is a Good Thing

How would I use the "hashed" password?

- You would validate, on login, that the password provided in the form matches the hashed password as follows:

```
<cfquery name="checkPwd" datasource="#Application.myDataSource#">
        SELECT Email, Password
        FROM User
        WHERE Email = <cfqueryparam cfsqltype="cf_sql_varchar" value="#Form.Email#">
</cfquery>

<cfif checkPwd.recordCount EQ 1>
        <cfif checkPwd.Pwd EQ Hash(form.password & variables.salt, "SHA-512")>
                <!--- Password is good – Woot --- Woot --- Log the person in --->
        <cfelse>
                <!--- Password is bad --->
        </cfif>
<cfelse>
        <!--- No E-Mail Match --->
</cfif>
```

Why SALTing Your Passwords is a Good Thing

References

- David Epler – Learn CF in a Week – Secure Password Storage: http://www.learncfinaweek.com/week1/Secure_Password_Storage/
- Generate Secret Key: https://helpx.adobe.com/coldfusion/cfml-reference/coldfusion-functions/functions-e-g/generatesecretkey.html or https://cfdocs.org/generatesecretkey
- Hash: https://helpx.adobe.com/coldfusion/cfml-reference/coldfusion-functions/functions-h-im/hash.html or https://cfdocs.org/hash

Upcoming ColdFusion Events

Adobe ColdFusion Summit 2018
Hard Rock Hotel and Casino – Las Vegas
October 1st – 3rd
Register at https://cfsummit.adobeevents.com/register/
$99.00 registration is still available until April 30th

Next Steps for the Seattle ColdFusion User Group

- Does this venue work for you, or would We Work in Bellevue work better?
- What would you like our group to focus on?
- What topics would you like to hear?

Volunteer Opportunities

- Social Media
- Web Site
- Presentations

Next Month’s Meeting

- May 9, 2018 – OnlineMetals.com