How to get your code deployed: Security

AMS Hackathon 2013
Chris Steipp
csteipp
<table>
<thead>
<tr>
<th>Security Awareness in the Community</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure Designs &amp; Designs that promote Security</td>
</tr>
<tr>
<td>Developers prevent flaws in their code</td>
</tr>
<tr>
<td>Codebase is analyzed for vulnerabilities</td>
</tr>
<tr>
<td>Response to reported vulnerabilities</td>
</tr>
</tbody>
</table>
Don't do these:

- Cross-site Scripting (XSS)
- Cross-site Request Forgery (CSRF)
- Register Globals
- SQL Injection

- XML External Entity (XXE) Processing
- Protect Confidential Information
XSS

- An attacker is able to inject client-side scripting into a web page viewed by other users

- Results in:
  - Attacker controls everything you do and see in your browser

- Types
  - Reflected
  - Stored
  - DOM
Examples

**Reflected XSS (1st Order)**

```html
<input type="text" name="search_term" value="<? echo $_GET['search_term']; ?>" />
```

**Stored XSS (2nd Order)**

```php
<?php
    $articles = $db->query("SELECT id, title FROM 'articles';");
    foreach ($articles as $article) {
        echo "<a href='read.php?id={$article['id']}'>{$article['title']}</a>";
    }
?>
```

**Dom-based XSS (3rd Order)**

```html
<script>
    document.write("<a href="'+document.referrer+'">Go Back</a>");
</script>
```
Best Practices

- Validate Input, Escape Output
- Trust No Input (including cookies, database, stuff you wrote in the dom)
- Use HTML/XML objects, know which functions escape and which don't
- Escape as close to the output as possible
- In javascript use: document.createElement(), element.setAttribute(), element.appendChild(); avoid element.html(), element.innerHTML(), document.href; avoid $ ("untrusted data")
- Always keep in mind the dom context where you are writing out user-controlled data!
$attrs = array(  
    'name' => 'wpSourceType',  
    'type' => 'radio',  
    'id' => $id,  
    'value' => $this-&gt;mParams['upload-type'],
);
if ( !empty( $this-&gt;mParams['checked'] ) ) {
    $attrs['checked'] = 'checked';
}
,label .= Html::element( 'input', $attrs );

$out .= Xml::openElement( 'div', array( 'class' => 'search-types' ) );
$out .= Xml::openElement( 'ul' );
...
$out .= Xml::closeElement( 'ul' );
$out .= Xml::closeElement( 'div' );
CSRF

- Attacker sends unauthorized commands from a user to a website that trusts that user.
- Abuses the way that web browsers send cookies with all requests to a domain, even when possibly initiated from another website.
CSRF

// a page on funnykitties.com
<img src='cat1.jpg' />
...
...

Or more likely:

<form name="wikiedit" method="POST"
    target="hiddenframe"
    <input type="hidden" name="wpTextbox1" value="whatever the attacker wants to say" />
    ...
</form>
<iframe name="hiddenframe" style="display: none"></iframe>
<script>
    document.wikiedit.submit();
</script>
CSRF Prevention

- Add a random token to forms, and check the form when processing it
  - If you parse a form, you should be using $user->matchEditToken() in your code
    - Login code has a slightly different token
  - Use api.php, and return true from needsToken();
  - Use 'private' group for ResourceLoader
  - Cookies are **not** csrf protection
  - Requiring POST is **not** csrf protection
Prevention

```php
$token = $request->getVal('wpEditToken');
$this->mTokenOk = $this->getUser()->matchEditToken($token);
```
Register Globals

- mostly not an issue (deprecated in php 5.4)
- But still make sure you don't introduce it in your code
SQL Injection

• $qry = "SELECT user_id, user_password FROM user WHERE user_name = '$userName' ";
Prevention

```php
$result = $dbw->select(
    'user', // table
    array( 'user_id', user_password ), // columns
    array( 'user_name' => $userName ), // where clause
    __METHOD__
);
```
Prevention

- use builders, but make sure you understand the functions!
Time for Exercise!

XXE

- Most XML processing libraries follow the XML standard, and allow resolving / including references in XML by URI
- http://, file://, expect://
- Can result in:
  - Server making requests, abusing network segmentation
  - Server including local files in output
  - Local code execution
XXE Prevention

- When processing XML with libxml2-based classes, disable external entity processing
  - libxml_disable_entity_loader( true );
Failure to Protect Confidential Data

• Some data on the wiki needs to be protected (privacy and legal reasons)

• Private Data: IP address, User Agent, Authentication Data
  − Don't store it unless you have to

• For legal compliance: Any contributed data can be deleted or suppressed
  − Usernames, Revisions, Page Titles, Images
Check if it's deleted

• Core
  - archive.ar_deleted
  - filearchive.fa_deleted
  - ipblocks.ipb_deleted
  - logging.log_deleted
  - revision.rev_deleted

• Common WMF Extensions
  - CentralAuth
    • globaluser.gu_hidden
  - Abuse Filter
    • ip/ua of logs (always)
    • abuse_filter.af_hidden
    • abuse_filter_log.afl_deleted
  - CheckUser
    • cu_log
Questions?