Fortify Developer Workbook

Jul 27, 2018

VHADURButtS

Report Overview

Report Summary

On Jul 26, 2018, a source code review was performed over the Inbound code base. 2,172 files, 36,047 LOC (Executable) were scanned. A total of 4 issues were uncovered during the analysis. This report provides a comprehensive description of all the types of issues found in this project. Specific examples and source code are provided for each issue type.

Issues by Fortify Priority Order	
High	4

Issue Summary

Overall number of results

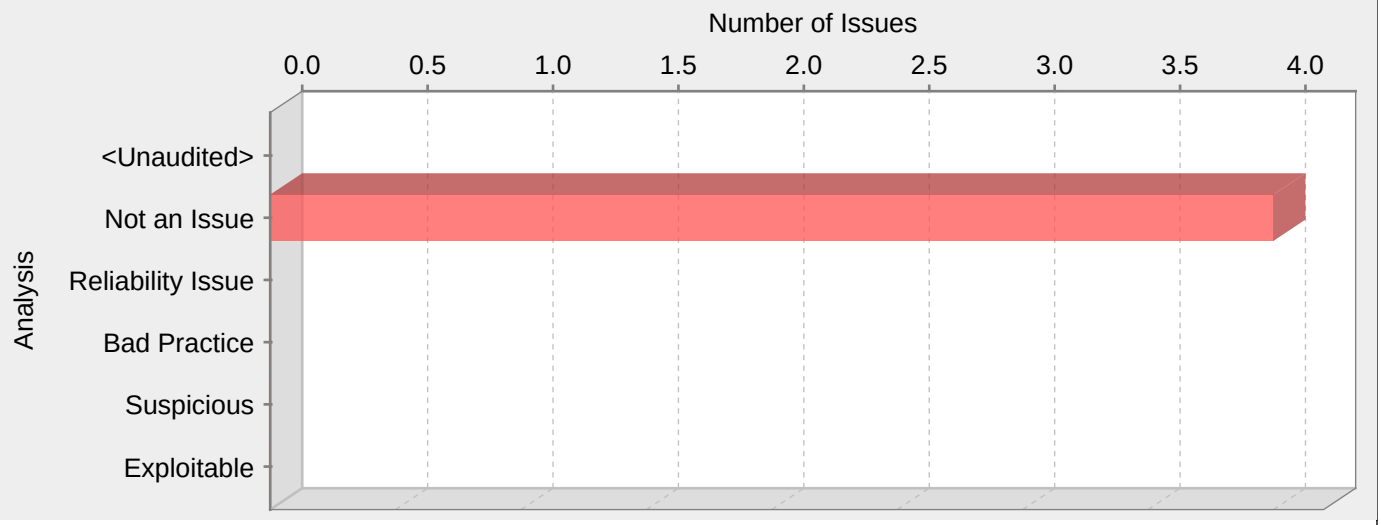
The scan found 4 issues.

Issues by Category	
Password Management: Hardcoded Password	4

Results Outline

Vulnerability Examples by Category

Category: Password Management: Hardcoded Password (4 Issues)



Abstract:

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

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. Once 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 will be forced to choose between security and availability.

Example 1: The following code uses a hardcoded password to connect to a database:

```
...
DriverManager.getConnection(url, "scott", "tiger");
...
```

This code will run successfully, but anyone who has access to it will have access to the password. Once the program has shipped, 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 could use it to break into the system. Even worse, if attackers have access to the bytecode for the application they can use the javap -c command to access the disassembled code, which will contain the values of the passwords used. The result of this operation might look something like the following for the example above:

```
javap -c ConnMngr.class
22: ldc   #36; //String jdbc:mysql://ixne.com/rxsql
24: ldc   #38; //String scott
26: ldc   #17; //String tiger
```

In the mobile world, password management is even trickier, considering a much higher chance of device loss.

Example 2: The code below uses hardcoded username and password to setup authentication for viewing protected pages with Android's WebView.

```
...
webview.setWebViewClient(new WebViewClient() {
public void onReceivedHttpAuthRequest(WebView view,
HttpAuthHandler handler, String host, String realm) {
handler.proceed("guest", "allow");
}
});
...
```

Similar to Example 1, this code will run successfully, but anyone who has access to it will have access to the password.

Recommendations:

Passwords should never be hardcoded and should generally be obfuscated and managed in an external source. Storing passwords in plaintext anywhere on the system allows anyone with sufficient permissions to read and potentially misuse the password. At the very least, passwords should be hashed before being stored.

Some third-party products claim the ability to manage passwords in a more secure way. For example, WebSphere Application Server 4.x uses a simple XOR encryption algorithm for obfuscating values, but be skeptical about such facilities. WebSphere and other application servers offer outdated and relatively weak encryption mechanisms that are insufficient for security-sensitive environments. For a secure generic solution, the best option today appears to be a proprietary mechanism that you create.

For Android, as well as any other platform that uses SQLite database, a good option is SQLCipher -- an extension to SQLite database that provides transparent 256-bit AES encryption of database files. Thus, credentials can be stored in an encrypted database.

Example 3: The code below demonstrates how to integrate SQLCipher into an Android application after downloading the necessary binaries, and store credentials into the database file.

```
import net.sqlcipher.database.SQLiteDatabase;
...
SQLiteDatabase.loadLibs(this);
File dbFile = getDatabasePath("credentials.db");
dbFile.mkdirs();
dbFile.delete();
SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(dbFile, "credentials", null);
db.execSQL("create table credentials(u, p)");
db.execSQL("insert into credentials(u, p) values(?, ?)", new Object[]{username, password});
...
```

Note that references to android.database.sqlite.SQLiteDatabase are substituted with those of net.sqlcipher.database.SQLiteDatabase.

To enable encryption on the WebView store, WebKit has to be re-compiled with the sqlcipher.so library.

Tips:

1. The Fortify Java Annotations FortifyPassword and FortifyNotPassword can be used to indicate which fields and variables represent passwords.

2. When identifying null, empty, or hardcoded passwords, default rules only consider fields and variables that contain the word password. However, the Fortify Custom Rules Editor provides the Password Management wizard that makes it easy to create rules for detecting password management issues on custom-named fields and variables.

OutboundNCPDPMessageServiceImpl.java, line 77 (Password Management: Hardcoded Password)

Fortify Priority:	High	**Folder**	High
Kingdom:	Security Features		
Abstract:	Hardcoded passwords may compromise system security in a way that cannot be easily remedied.		
Sink:	OutboundNCPDPMessageServiceImpl.java:77 FieldAccess: KEYSTORE_PASSWD_PROPERTY()		

```
75    private static final String KEYSTORE_FILE_TYPE = "JKS";
76    private static final String KEYSTORE_FILE_NAME_PROPERTY = "keystore.filename";
77    private static final String KEYSTORE_PASSWD_PROPERTY = "keystore.password";
78
79    @Autowired
```

Analysis:	Not an Issue	
Comments:	*VHAHAMWandlJ 2017-05-10 11:28 AM*	The references are not hardcoded passwords, but System Property names that refer to passwords

ClientPasswordCallback.java, line 18 (Password Management: Hardcoded Password)

Fortify Priority:	High	**Folder**	High
Kingdom:	Security Features		
Abstract:	Hardcoded passwords may compromise system security in a way that cannot be easily remedied.		
Sink:	ClientPasswordCallback.java:18 FieldAccess: E_AND_E_PASSWD()		

```
16
17    private static final String WSCLIENTS_PROPERTIES_FILE =
"gov.va.med.pharmacy.inboundeRx.properties";
```

```
18    private static final String E_AND_E_PASSWD = "eAnde.password";
19    private static final String E_AND_E_USERNAME = "eAnde.username";
```

Analysis:	Not an Issue
Comments:	*VHAHAMWandlJ 2017-05-10 11:28 AM* The references are not hardcoded passwords, but System Property names that refer to passwords

EESummary_Client.java, line 55 (Password Management: Hardcoded Password)

Fortify Priority:	High	**Folder**	High
Kingdom:	Security Features		
Abstract:	Hardcoded passwords may compromise system security in a way that cannot be easily remedied.		
Sink:	EESummary_Client.java:55 FieldAccess: KEYSTORE_PASSWD_PROPERTY()		

```
53    private static final String KEYSTORE_FILE_TYPE = "JKS";
54    private static final String KEYSTORE_FILE_NAME_PROPERTY = "keystore.filename";
55    private static final String KEYSTORE_PASSWD_PROPERTY = "keystore.password";
56
57        public getEESummaryResponse returnResponse(String key) throws java.lang.Exception
{
```

Analysis:	Not an Issue
Comments:	*VHAHAMWandlJ 2017-09-14 4:55 PM* The references are not hardcoded passwords, but System Property names that refer to passwords

MVIClient.java, line 97 (Password Management: Hardcoded Password)

Fortify Priority:	High	**Folder**	High
Kingdom:	Security Features		
Abstract:	Hardcoded passwords may compromise system security in a way that cannot be easily remedied.		
Sink:	MVIClient.java:97 FieldAccess: KEYSTORE_PASSWD_PROPERTY()		

```
95    private static final String KEYSTORE_FILE_TYPE = "JKS";
96    private static final String KEYSTORE_FILE_NAME_PROPERTY = "keystore.filename";
97    private static final String KEYSTORE_PASSWD_PROPERTY = "keystore.password";
98    private static final String LEGAL_NAME = "Legal Name";
99    private static final String L_CHARACTER = "L";
```

Analysis:	Not an Issue
Comments:	*VHAHAMWandlJ 2017-05-10 11:28 AM* The references are not hardcoded passwords, but System Property names that refer to passwords