Hewlett Packard Enterprise

HPE Security Fortify Audit Workbench

Developer Workbook

PS_INB_ERX_Fortify-2.0.4.057

FORTIFY

Table of Contents

© Copyright 2016 Hewlett Packard Enterprise Development LP

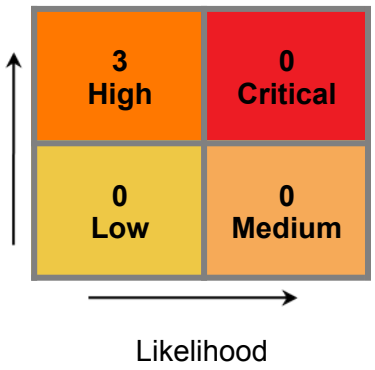
Executive Summary

This workbook is intended to provide all necessary details and information for a developer to understand and remediate the different issues discovered during the PS_INB_ERX_Fortify-2.0.4.057 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:	PS_INB_ERX_Fortify-2.(
Project Version:	
SCA:	Results Present
WebInspect:	Results Not Present
SecurityScope:	Results Not Present
Other:	Results Not Present

Issues by Priority



Top Ten Critical Categories

This project does not contain any critical issues

© Copyright 2016 Hewlett Packard Enterprise Development LP

Project Description

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

SCA

Date of Last Analysis:	Jul 25, 2017, 1:17 PM	**Engine Version:**	17.10.0156
Host Name:	HAM-LT51592	**Certification:**	VALID
Number of Files:	2,150	**Lines of Code:**	108,904

© Copyright 2016 Hewlett Packard Enterprise Development LP

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	
Password Management: Hardcoded Password	0	3 / 3	0	0	3 / 3

© Copyright 2016 Hewlett Packard Enterprise Development LP

Results Outline

Password Management: Hardcoded Password (3 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");
  }
});
...
```


© Copyright 2016 Hewlett Packard Enterprise Development LP

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

Recommendation

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.

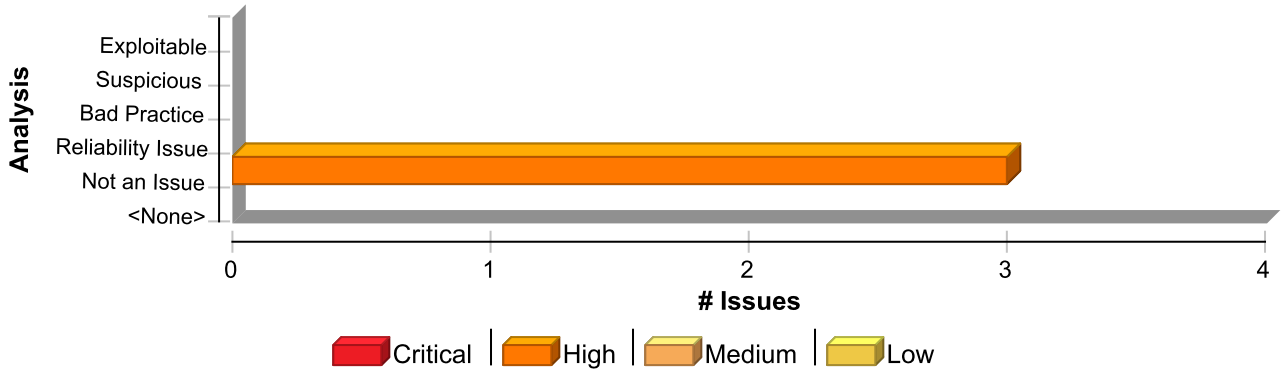
```
import net.sqlcipher.database.SQLiteDatabase;
...
  SQLiteDatabase.loadLibs(this);
  File dbFile = getDatabasePath("credentials.db");
  dbFile.mkdirs();
  dbFile.delete();
  SQLiteDatabase db = SQLiteDatabase.openOrCreateDatabase(dbFile,
"credentials", null);
  db.execSQL("create table credentials(u, 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.

Issue Summary

© Copyright 2016 Hewlett Packard Enterprise Development LP



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Hardcoded Password	3	0	0	3
Total	**3**	**0**	**0**	**3**

Password Management: Hardcoded Password — High

Package: gov.va.med.pharmacy.jaxrs.mvi.dao.impl

PS_INB_ERX_WS/src/main/java/gov/va/med/pharmacy/jaxrs/mvi/dao/impl/MVIClient.java, line 98 (Password Management: Hardcoded Password) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

VHAHAMWandlJ: Wed May 10 2017 11:28:37 GMT-0400 (EDT)
The references are not hardcoded passwords, but System Property names that refer to passwords

Sink Details

Sink: FieldAccess: KEYSTORE_PASSWD_PROPERTY
Enclosing Method: ()
File: PS_INB_ERX_WS/src/main/java/gov/va/med/pharmacy/jaxrs/mvi/dao/impl/MVIClient.java:98
Taint Flags:

```
95  private static final String MVI_CLIENT_ENDPOINT_PROPERTY = "mvi.client.endpoint";
96  private static final String KEYSTORE_FILE_TYPE = "JKS";
97  private static final String KEYSTORE_FILE_NAME_PROPERTY = "keystore.filename";
98  private static final String KEYSTORE_PASSWD_PROPERTY = "keystore.password";
99  private static final String LEGAL_NAME = "Legal Name";
100  private static final String L_CHARACTER = "L";
101  private static final String EMPTY_STRING = "";
```


© Copyright 2016 Hewlett Packard Enterprise Development LP

Password Management: Hardcoded Password	High

Package: gov.va.med.pharmacy.jaxrs.outboundncpdpmessage.service.impl

PS_INB_ERX_WS/src/main/java/gov/va/med/pharmacy/jaxrs/outboundncpdpmessage/service/impl/OutboundNCPDPMessageServiceImpl.java, line 76 (Password Management: Hardcoded Password)	High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

VHAHAMWandlJ: Wed May 10 2017 11:28:46 GMT-0400 (EDT)
The references are not hardcoded passwords, but System Property names that refer to passwords

Sink Details

Sink: FieldAccess: KEYSTORE_PASSWD_PROPERTY
Enclosing Method: ()
File: PS_INB_ERX_WS/src/main/java/gov/va/med/pharmacy/jaxrs/outboundncpdpmessage/service/impl/OutboundNCPDPMessageServiceImpl.java:76
Taint Flags:

```
73
74  private static final String KEYSTORE_FILE_TYPE = "JKS";
75  private static final String KEYSTORE_FILE_NAME_PROPERTY = "keystore.filename";
76  private static final String KEYSTORE_PASSWD_PROPERTY = "keystore.password";
77
78  @Autowired
79  private OutboundNcpdpMsgService outboundNcpdpMsgService;
```

Package: gov.va.med.pharmacy.wsclients.eAnde

PS_INB_ERX_Common/src/main/java/gov/va/med/pharmacy/wsclients/eAnde/ClientPasswordCallback.java, line 17 (Password Management: Hardcoded Password)	High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Audit Details

Analysis Not an Issue

Audit Comments

VHAHAMWandlJ: Wed May 10 2017 11:28:30 GMT-0400 (EDT)
The references are not hardcoded passwords, but System Property names that refer to passwords

Sink Details

Sink: FieldAccess: E_AND_E_PASSWD
Enclosing Method: ()
File: PS_INB_ERX_Common/src/main/java/gov/va/med/pharmacy/wsclients/eAnde/ClientPasswordCallback.java:17
Taint Flags:

© Copyright 2016 Hewlett Packard Enterprise Development LP

Password Management: Hardcoded Password	High
Package: gov.va.med.pharmacy.wsclients.eAnde	
PS_INB_ERX_Common/src/main/java/gov/va/med/pharmacy/wsclients/eAnde/ClientPasswordCallback.java, line 17 (Password Management: Hardcoded Password)	High

```
14
15 public class ClientPasswordCallback implements CallbackHandler {
16
17  private static final String E_AND_E_PASSWD = "eAnde.password";
18  private static final String E_AND_E_USERNAME = "eAnde.username";
19
20  public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
```


© Copyright 2016 Hewlett Packard Enterprise Development LP

© Copyright 2016 Hewlett Packard Enterprise Development LP