Hewlett Packard Enterprise

HPE Security Fortify Audit Workbench

Developer Workbook

ecrnow

FORTIFY®

Table of Contents

Executive Summary
Project Description
Issue Breakdown by Fortify Categories
Results Outline
About HPE Security Enterprise Security Products

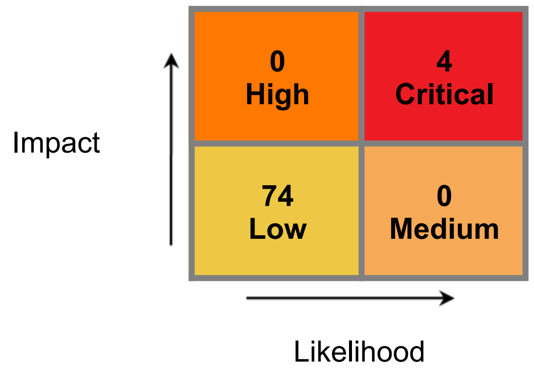
Executive Summary

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

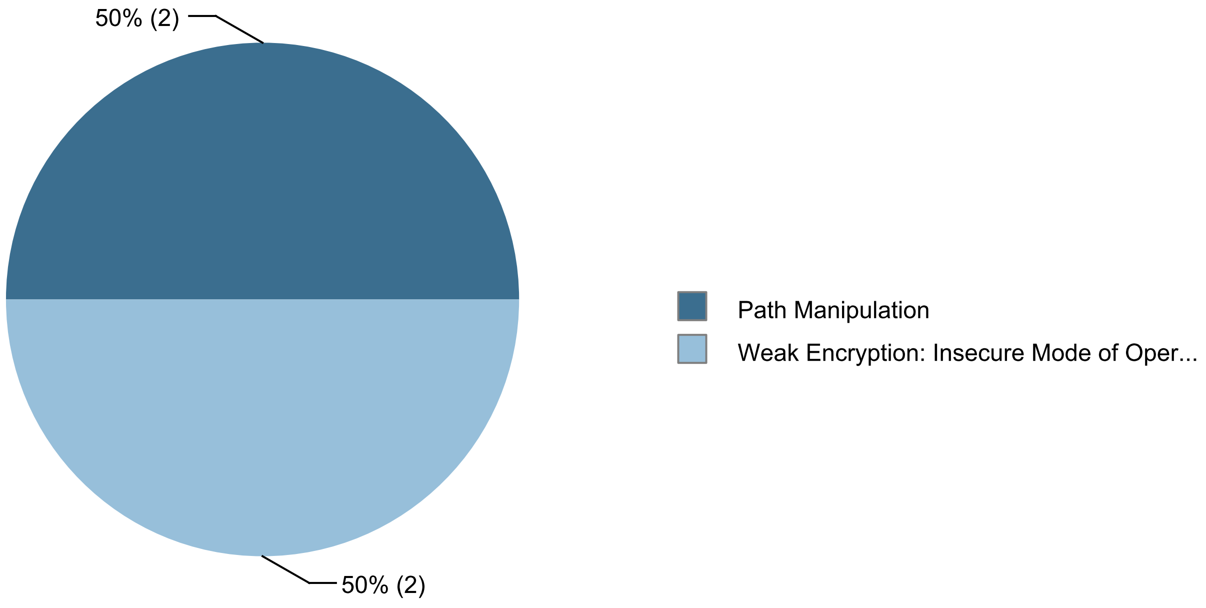
This section provides an overview of the issues uncovered during analysis.

Project Name:	ecrnow
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 HPE Security Fortify scan engines used for this project, as well as the project meta-information.

SCA

Date of Last Analysis:	Dec 10, 2020, 9:21 AM	**Engine Version:**	17.20.0183
Host Name:	W1970528	**Certification:**	VALID
Number of Files:	134	**Lines of Code:**	15,653

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	
Missing Check against Null	0	0	0	0 / 1	0 / 1
Path Manipulation	0 / 2	0	0	0	0 / 2
Poor Error Handling: Overly Broad Catch	0	0	0	0 / 54	0 / 54
Poor Error Handling: Overly Broad Throws	0	0	0	0 / 6	0 / 6
Redundant Null Check	0	0	0	0 / 8	0 / 8
System Information Leak	0	0	0	0 / 2	0 / 2
System Information Leak: Internal	0	0	0	0 / 2	0 / 2
Weak Cryptographic Hash	0	0	0	0 / 1	0 / 1
Weak Encryption: Insecure Mode of Operation	0 / 2	0	0	0	0 / 2

Results Outline

Missing Check against Null (1 issue)

Abstract

The program can dereference a null pointer because it does not check the return value of a function that might return null.

Explanation

Just about every serious attack on a software system begins with the violation of a programmer's assumptions. After the attack, the programmer's assumptions seem flimsy and poorly founded, but before an attack many programmers would defend their assumptions well past the end of their lunch break. Two dubious assumptions that are easy to spot in code are "this function call can never fail" and "it doesn't matter if this function call fails". When a programmer ignores the return value from a function, they implicitly state that they are operating under one of these assumptions. **Example 1:** The following code does not check to see if the string returned by `getParameter()` is `null` before calling the member function `compareTo()`, potentially causing a null dereference.

```
String itemName = request.getParameter(ITEM_NAME);
    if (itemName.compareTo(IMPORTANT_ITEM)) {
        ...
    }
    ...
```

Example 2:. The following code shows a system property that is set to `null` and later dereferenced by a programmer who mistakenly assumes it will always be defined.

```
System.clearProperty("os.name");
...
String os = System.getProperty("os.name");
if (os.equalsIgnoreCase("Windows 95") )
    System.out.println("Not supported");
```

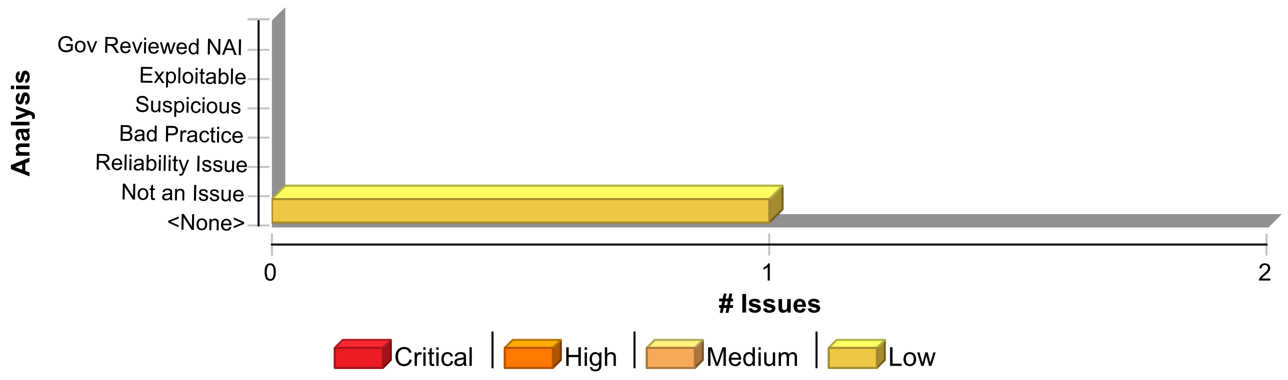
The traditional defense of this coding error is: "I know the requested value will always exist because.... If it does not exist, the program cannot perform the desired behavior so it doesn't matter whether I handle the error or simply allow the program to die dereferencing a null value." But attackers are skilled at finding unexpected paths through programs, particularly when exceptions are involved.

Recommendation

If a function can return an error code or any other evidence of its success or failure, always check for the error condition, even if there is no obvious way for it to occur. In addition to preventing security errors, many initially mysterious bugs have eventually led back to a failed method call with an unchecked return value. Create an easy to use and standard way for dealing with failure in your application. If error handling is straightforward, programmers will be less inclined to omit it. One approach to standardized error handling is to write wrappers around commonly-used functions that check and handle error conditions without additional programmer intervention. When wrappers are implemented and adopted, the use of non-wrapped equivalents can be prohibited and enforced by using custom rules. **Example 3:** The following code implements a wrapper around `getParameter()` that checks the return value of `getParameter()` against `null` and uses a default value if the requested parameter is not defined.

```
String safeGetParameter (HttpRequest request, String name)
{
    String value = request.getParameter(name);
    if (value == null) {
        return getDefaultValue(name)
    }
    return value;
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Missing Check against Null	1	0	0	1
Total	**1**	**0**	**0**	**1**

Missing Check against Null — Low

Package: com.drajer.cda.utils

cda/utils/CdaGeneratorConstants.java, line 853 (Missing Check against Null) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Control Flow)

Sink Details

Sink: getClassLoader() : Class.getClassLoader may return NULL
Enclosing Method: ()
File: cda/utils/CdaGeneratorConstants.java:853
Taint Flags:

```
850  static {
851  try (InputStream input =
852  CdaGeneratorConstants.class
853  .getClassLoader()
854  .getResourceAsStream("oid-uri-mapping-r4.properties")) {
855  Properties prop = new Properties();
856  prop.load(input);
```

Path Manipulation (2 issues)

Abstract

Allowing user input to control paths used in file system operations could enable an attacker to access or modify otherwise protected system resources.

Explanation

Path manipulation errors occur when the following two conditions are met: 1. An attacker is able to specify a path used in an operation on the file system. 2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted. For example, the program may give the attacker the ability to overwrite the specified file or run with a configuration controlled by the attacker. **Example 1:** The following code uses input from an HTTP request to create a file name. The programmer has not considered the possibility that an attacker could provide a file name such as "`../../tomcat/conf/server.xml`", which causes the application to delete one of its own configuration files.

```
String rName = request.getParameter("reportName");
File rFile = new File("/usr/local/apfr/reports/" + rName);
...
rFile.delete();
```

Example 2: The following code uses input from a configuration file to determine which file to open and echo back to the user. If the program runs with adequate privileges and malicious users can change the configuration file, they can use the program to read any file on the system that ends with the extension `.txt`.

```
fis = new FileInputStream(cfg.getProperty("sub")+".txt");
amt = fis.read(arr);
out.println(arr);
```

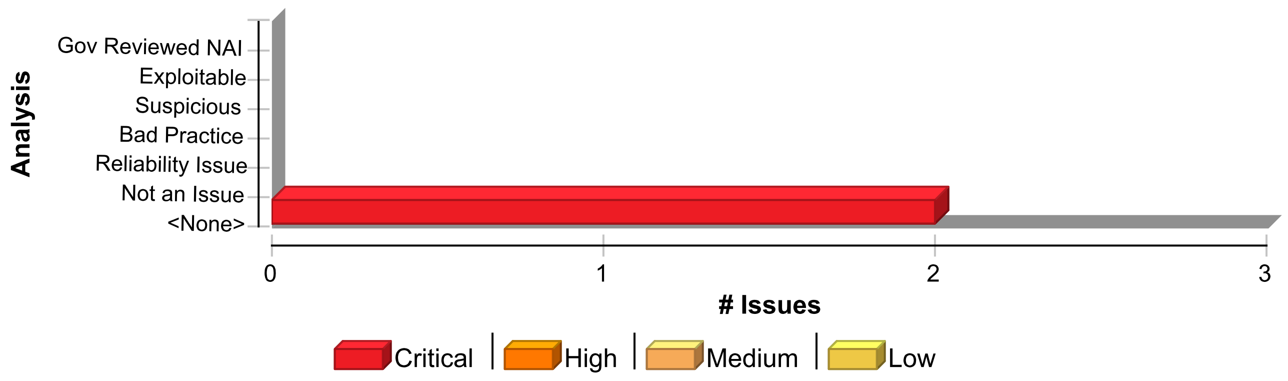
Some think that in the mobile world, classic vulnerabilities, such as path manipulation, do not make sense -- why would the user attack themself? However, keep in mind that the essence of mobile platforms is applications that are downloaded from various sources and run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which necessitates expanding the attack surface of mobile applications to include inter-process communication. **Example 3:** The following code adapts Example 1 to the Android platform.

```
...
        String rName = this.getIntent().getExtras().getString("reportName");
        File rFile = getBaseContext().getFileStreamPath(rName);
...
        rFile.delete();
...
```

Recommendation

The best way to prevent path manipulation is with a level of indirection: create a list of legitimate resource names that a user is allowed to specify, and only allow the user to select from the list. With this approach the input provided by the user is never used directly to specify the resource name. In some situations this approach is impractical because the set of legitimate resource names is too large or too hard to keep track of. Programmers often resort to blacklisting in these situations. Blacklisting selectively rejects or escapes potentially dangerous characters before using the input. However, any such list of unsafe characters is likely to be incomplete and will almost certainly become out of date. A better approach is to create a whitelist of characters that are allowed to appear in the resource name and accept input composed exclusively of characters in the approved set.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Path Manipulation	2	0	0	2
Total	**2**	**0**	**0**	**2**

Path Manipulation — Critical

Package: com.drajer.routing.impl

routing/impl/DirectResponseReceiver.java, line 110 (Path Manipulation) — Critical

Issue Details

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

Source Details

Source: javax.mail.Store.getFolder()
From: com.drajer.routing.impl.DirectResponseReceiver.readMail
File: routing/impl/DirectResponseReceiver.java:75

```
72  logger.info("Connecting to IMAP Inbox");
73  store.connect(details.getDirectHost(), port, details.getDirectUser(),
details.getDirectPwd());
74
75  Folder inbox = store.getFolder(INBOX);
76  inbox.open(Folder.READ_WRITE);
77
78  Flags seen = new Flags(Flags.Flag.SEEN);
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: readMail()
File: routing/impl/DirectResponseReceiver.java:110
Taint Flags: NETWORK, XSS

```
107
108  try (InputStream stream = bodyPart.getInputStream()) {
109  byte[] targetArray = IOUtils.toByteArray(stream);
110  FileUtils.writeByteArrayToFile(new File(filename), targetArray);
```

Path Manipulation — Critical

Package: com.drajer.routing.impl

routing/impl/DirectResponseReceiver.java, line 110 (Path Manipulation) — Critical

```
111  }
112  File file1 = new File(filename);
113  FileBody fileBody = new FileBody(file1);
```

routing/impl/DirectResponseReceiver.java, line 112 (Path Manipulation) — Critical

Issue Details

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

Source Details

Source: javax.mail.Store.getFolder()
From: com.drajer.routing.impl.DirectResponseReceiver.readMail
File: routing/impl/DirectResponseReceiver.java:75

```
72  logger.info("Connecting to IMAP Inbox");
73  store.connect(details.getDirectHost(), port, details.getDirectUser(),
details.getDirectPwd());
74
75  Folder inbox = store.getFolder(INBOX);
76  inbox.open(Folder.READ_WRITE);
77
78  Flags seen = new Flags(Flags.Flag.SEEN);
```

Sink Details

Sink: java.io.File.File()
Enclosing Method: readMail()
File: routing/impl/DirectResponseReceiver.java:112
Taint Flags: NETWORK, XSS

```
109  byte[] targetArray = IOUtils.toByteArray(stream);
110  FileUtils.writeByteArrayToFile(new File(filename), targetArray);
111  }
112  File file1 = new File(filename);
113  FileBody fileBody = new FileBody(file1);
114
115  logger.info(
```

Poor Error Handling: Overly Broad Catch (54 issues)

Abstract

The catch block handles a broad swath of exceptions, potentially trapping dissimilar issues or problems that should not be dealt with at this point in the program.

Explanation

Multiple catch blocks can get repetitive, but "condensing" catch blocks by catching a high-level class such as `Exception` can obscure exceptions that deserve special treatment or that should not be caught at this point in the program. Catching an overly broad exception essentially defeats the purpose of Java's typed exceptions, and can become particularly dangerous if the program grows and begins to throw new types of exceptions. The new exception types will not receive any attention. **Example:** The following code excerpt handles three types of exceptions in an identical fashion.

```
  try {
    doExchange();
  }
  catch (IOException e) {
    logger.error("doExchange failed", e);
  }
  catch (InvocationTargetException e) {
    logger.error("doExchange failed", e);
  }
  catch (SQLException e) {
    logger.error("doExchange failed", e);
  }
```

At first blush, it may seem preferable to deal with these exceptions in a single catch block, as follows:

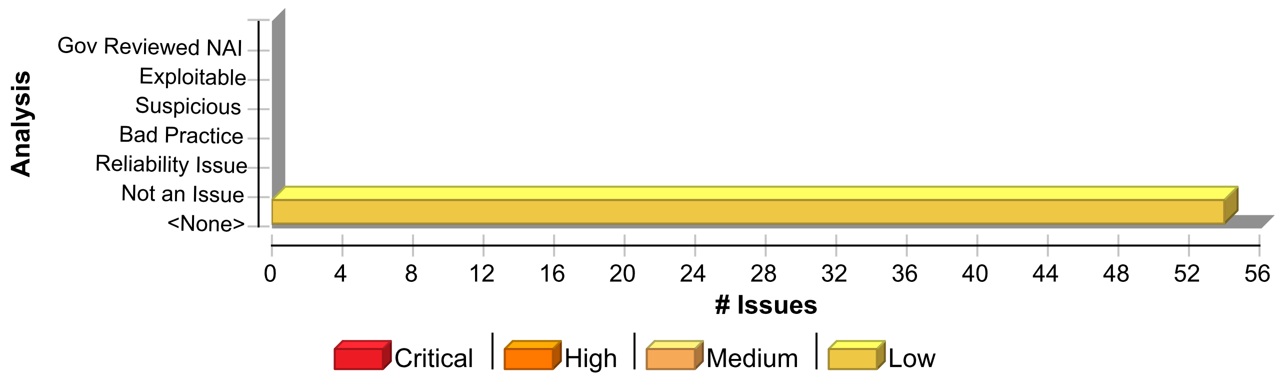
```
  try {
    doExchange();
  }
  catch (Exception e) {
    logger.error("doExchange failed", e);
  }
```

However, if `doExchange()` is modified to throw a new type of exception that should be handled in some different kind of way, the broad catch block will prevent the compiler from pointing out the situation. Further, the new catch block will now also handle exceptions derived from `RuntimeException` such as `ClassCastException`, and `NullPointerException`, which is not the programmer's intent.

Recommendation

Do not catch broad exception classes such as `Exception`, `Throwable`, `Error`, or `RuntimeException` except at the very top level of the program or thread.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Overly Broad Catch	54	0	0	54
Total	**54**	**0**	**0**	**54**

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.cda.utils

cda/utils/CdaValidatorUtil.java, line 106 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: validateEicrToSchematron()
File: cda/utils/CdaValidatorUtil.java:106
Taint Flags:

```
logger.info("Found Valid Schematron which can be applied EICR ");
output =
aResSCH.applySchematronValidationToSVRL(new StreamSource(new StringReader(ecrData)));
} catch (Exception e) {
logger.error("Unable to read/write execution state: " + e.getMessage());
}

```

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 58 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 58 (Poor Error Handling: Overly Broad Catch) — Low

Sink: CatchBlock
Enclosing Method: decrypt()
File: ecrapp/security/AESEncryption.java:58
Taint Flags:

```
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
cipher.init(Cipher.DECRYPT_MODE, secretKey);
return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
} catch (Exception e) {
System.out.println("Error while decrypting: " + e.toString());
}
return null;
```

ecrapp/security/AESEncryption.java, line 46 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: encrypt()
File: ecrapp/security/AESEncryption.java:46
Taint Flags:

```
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secretKey);
return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
} catch (Exception e) {
System.out.println("Error while encrypting: " + e.toString());
}
return null;
```

Package: com.drajer.ecrapp.service

ecrapp/service/PlanDefinitionProcessor.java, line 382 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: readErsdBundleFromFile()
File: ecrapp/service/PlanDefinitionProcessor.java:382

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.ecrapp.service

ecrapp/service/PlanDefinitionProcessor.java, line 382 (Poor Error Handling: Overly Broad Catch) — Low

Taint Flags:

```
bundle = jsonParser.parseResource(Bundle.class, in);
logger.info("Completed Reading ERSD File");
} catch (Exception e) {
logger.error("Exception Reading ERSD File", e);
}
return bundle;
```

Package: com.drajer.ecrapp.util

ecrapp/util/ApplicationUtils.java, line 455 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: readBundleFromFile()
File: ecrapp/util/ApplicationUtils.java:455
Taint Flags:

```
bundle = jsonParser.parseResource(Bundle.class, in);
logger.info("Completed Reading File");
} catch (Exception e) {
logger.error("Exception Reading File", e);
}
return bundle;
```

Package: com.drajer.routing

routing/FhirEicrSender.java, line 52 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: submitBundle()
File: routing/FhirEicrSender.java:52
Taint Flags:

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.routing

routing/FhirEicrSender.java, line 52 (Poor Error Handling: Overly Broad Catch) — Low

```
logger.info("Saving response to file::::{}", fileName);
ApplicationUtils.saveDataToFile(response.getBody(), fileName);

} catch (Exception e) {
logger.error("Error in Submitting Bundle to FHIR Endpoint: " + fhirServerURL);
}
return bundleResponse;
```

routing/RestApiSender.java, line 73 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: sendEicrXmlDocument()
File: routing/RestApiSender.java:73
Taint Flags:

```
logger.info("Received response: {}", bundleResponse.toString());
}

} catch (Exception e) {

if (ub != null) {
logger.error("Error in Sending Eicr XML to Endpoint: {}", ub.toString());
```

Package: com.drajer.routing.impl

routing/impl/DirectResponseReceiver.java, line 126 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: readMail()
File: routing/impl/DirectResponseReceiver.java:126
Taint Flags:

```

deleteMail(details.getDirectHost(), details.getDirectUser(), details.getDirectPwd());

} catch (Exception e) {
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.routing.impl

routing/impl/DirectResponseReceiver.java, line 126 (Poor Error Handling: Overly Broad Catch) — Low

```

logger.error("Error while reading mail", e);
}
```

Package: com.drajer.sof.launch

sof/launch/LaunchController.java, line 323 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: launchApp()
File: sof/launch/LaunchController.java:323
Taint Flags:

```
320 // response.setStatus(HttpServletResponse.SC_TEMPORARY_REDIRECT);
321 // response.setHeader("Location", constructedAuthUrl);
322 }
323 } catch (Exception e) {
324 logger.error("Error in getting Authorization with Server");
325 }
326 } else {
```

Package: com.drajer.sof.service

sof/service/TriggerQueryDstu2Bundle.java, line 66 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/TriggerQueryDstu2Bundle.java:66
Taint Flags:

```
63 Entry patientEntry = new Entry();
64 patientEntry.setResource(patient);
65 bundle.addEntry(patientEntry);
66 } catch (Exception e) {
67 logger.error("Error in getting Patient Data");
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/TriggerQueryDstu2Bundle.java, line 66 (Poor Error Handling: Overly Broad Catch) — Low

```
68  }
69  // Step 1: Get Encounters for Patient based on encId. (Create a method to get
```

sof/service/TriggerQueryDstu2Bundle.java, line 133 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/TriggerQueryDstu2Bundle.java:133
Taint Flags:

```
130  }
131  Entry encounterEntry = new Entry().setResource(encounter);
132  bundle.addEntry(encounterEntry);
133  } catch (Exception e) {
134  logger.error("Error in getting Encounter Data");
135  }
136
```

sof/service/TriggerQueryDstu2Bundle.java, line 156 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/TriggerQueryDstu2Bundle.java:156
Taint Flags:

```
153  Entry conditionsEntry = new Entry().setResource(condition);
154  bundle.addEntry(conditionsEntry);
155  }
156  } catch (Exception e) {
157  logger.error("Error in getting Condition Data");
158  }
159
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/TriggerQueryDstu2Bundle.java, line 178 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/TriggerQueryDstu2Bundle.java:178
Taint Flags:

```
175  Entry observationsEntry = new Entry().setResource(observation);
176  bundle.addEntry(observationsEntry);
177  }
178  } catch (Exception e) {
179  logger.error("Error in getting Observation Data");
180  }
181
```

sof/service/TriggerQueryDstu2Bundle.java, line 234 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/TriggerQueryDstu2Bundle.java:234
Taint Flags:

```
231  Entry medAdministrationEntry = new Entry().setResource(medAdministration);
232  bundle.addEntry(medAdministrationEntry);
233  }
234  } catch (Exception e) {
235  logger.error("Error in getting the MedicationAdministration Data");
236  }
237
```

sof/service/TriggerQueryDstu2Bundle.java, line 258 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/TriggerQueryDstu2Bundle.java, line 258 (Poor Error Handling: Overly Broad Catch) — Low

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/TriggerQueryDstu2Bundle.java:258
Taint Flags:

```
Entry diagnosticOrderEntry = new Entry().setResource(diagnosticOrder);
bundle.addEntry(diagnosticOrderEntry);
}
} catch (Exception e) {
logger.error("Error in getting the DiagnosticOrder Data");
}

```

sof/service/TriggerQueryDstu2Bundle.java, line 272 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/TriggerQueryDstu2Bundle.java:272
Taint Flags:

```
Entry diagnosticReportEntry = new Entry().setResource(diagnosticReport);
bundle.addEntry(diagnosticReportEntry);
}
} catch (Exception e) {
logger.error("Error in getting the DiagnosticReport Data");
}

```

sof/service/LoadingQueryService.java, line 35 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getData()
File: sof/service/LoadingQueryService.java:35

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryService.java, line 35 (Poor Error Handling: Overly Broad Catch) — Low

Taint Flags:

```
Bundle bundle = new Bundle();
try {
bundle = generateDSTU2Bundle.createDSTU2Bundle(launchDetails, dstu2FhirData, start, end);
} catch (Exception e) {
logger.error("Error in Generating the DSTU2 Bundle");
}
dstu2FhirData.setData(bundle);
```

sof/service/LoadingQueryService.java, line 46 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getData()
File: sof/service/LoadingQueryService.java:46
Taint Flags:

```
org.hl7.fhir.r4.model.Bundle bundle = new org.hl7.fhir.r4.model.Bundle();
try {
bundle = generateR4Bundle.createR4Bundle(launchDetails, r4FhirData, start, end);
} catch (Exception e) {
logger.error("Error in Generating the R4 Bundle");
}
r4FhirData.setData(bundle);
```

sof/service/TriggerQueryService.java, line 35 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getData()
File: sof/service/TriggerQueryService.java:35
Taint Flags:

```
Bundle bundle = new Bundle();
try {
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/TriggerQueryService.java, line 35 (Poor Error Handling: Overly Broad Catch) — Low

```
bundle = generateDstu2Bundles.createDSTU2Bundle(launchDetails, dstu2FhirData, start, end);
} catch (Exception e) {
logger.error("Error in Generating the DSTU2 Bundle");
}
dstu2FhirData.setData(bundle);
```

sof/service/TriggerQueryService.java, line 49 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getData()
File: sof/service/TriggerQueryService.java:49
Taint Flags:

```
org.hl7.fhir.r4.model.Bundle bundle = new org.hl7.fhir.r4.model.Bundle();
try {
bundle = generateR4Bundles.createR4Bundle(launchDetails, r4FhirData, start, end);
} catch (Exception e) {
logger.error("Error in Generating the R4 Bundle");
}
r4FhirData.setData(bundle);
```

sof/service/LoadingQueryR4Bundle.java, line 65 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createR4Bundle()
File: sof/service/LoadingQueryR4Bundle.java:65
Taint Flags:

```
new BundleEntryComponent().setResource(observation);
bundle.addEntry(observationsEntry);
}
} catch (Exception e) {
logger.error("Error in getting Pregnancy Observation Data - {}, ", e, e);
}
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryR4Bundle.java, line 65 (Poor Error Handling: Overly Broad Catch) — Low

```
68
```

sof/service/LoadingQueryR4Bundle.java, line 84 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createR4Bundle()
File: sof/service/LoadingQueryR4Bundle.java:84
Taint Flags:

```
81  new BundleEntryComponent().setResource(observation);
82  bundle.addEntry(observationsEntry);
83  }
84  } catch (Exception e) {
85  logger.error("Error in getting Travel Observation Data - {}, ", e, e);
86  }
87
```

sof/service/LoadingQueryR4Bundle.java, line 103 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createR4Bundle()
File: sof/service/LoadingQueryR4Bundle.java:103
Taint Flags:

```
100  new BundleEntryComponent().setResource(observation);
101  bundle.addEntry(observationsEntry);
102  }
103  } catch (Exception e) {
104  logger.error("Error in getting Social History Observation(Occupation) Data");
105  }
106
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryR4Bundle.java, line 121 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createR4Bundle()
File: sof/service/LoadingQueryR4Bundle.java:121
Taint Flags:

```
BundleEntryComponent conditionEntry = new BundleEntryComponent().setResource(condition);
bundle.addEntry(conditionEntry);
}
} catch (Exception e) {
logger.error("Error in getting Pregnancy Conditions");
}

```

sof/service/LoadingQueryR4Bundle.java, line 139 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createR4Bundle()
File: sof/service/LoadingQueryR4Bundle.java:139
Taint Flags:

```
new BundleEntryComponent().setResource(medStatement);
bundle.addEntry(medStatementEntry);
}
} catch (Exception e) {
logger.error("Error in getting the MedicationStatement Data - {}, ", e, e);
}

```

sof/service/LoadingQueryR4Bundle.java, line 162 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryR4Bundle.java, line 162 (Poor Error Handling: Overly Broad Catch) — Low

Sink Details

Sink: CatchBlock
Enclosing Method: createR4Bundle()
File: sof/service/LoadingQueryR4Bundle.java:162
Taint Flags:

```
159 new BundleEntryComponent().setResource(immunization);
160 bundle.addEntry(immunizationEntry);
161 }
162 } catch (Exception e) {
163 logger.error("Error in getting the Immunization Data - {}, ", e, e);
164 }
165
```

sof/service/LoadingQueryR4Bundle.java, line 185 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createR4Bundle()
File: sof/service/LoadingQueryR4Bundle.java:185
Taint Flags:

```
182 new BundleEntryComponent().setResource(diagnosticReport);
183 bundle.addEntry(diagnosticReportEntry);
184 }
185 } catch (Exception e) {
186 logger.error("Error in getting the DiagnosticReport Data - {}, ", e, e);
187 }
188
```

sof/service/LoadingQueryDstu2Bundle.java, line 67 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:67

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryDstu2Bundle.java, line 67 (Poor Error Handling: Overly Broad Catch) — Low

Taint Flags:

```
Entry patientEntry = new Entry();
patientEntry.setResource(patient);
bundle.addEntry(patientEntry);
} catch (Exception e) {
logger.error("Error in getting Patient Data");
}

```

sof/service/LoadingQueryDstu2Bundle.java, line 135 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:135
Taint Flags:

```
}
Entry encounterEntry = new Entry().setResource(encounter);
bundle.addEntry(encounterEntry);
} catch (Exception e) {
logger.error("Error in getting Encounter Data");
}

```

sof/service/LoadingQueryDstu2Bundle.java, line 158 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:158
Taint Flags:

```
Entry conditionsEntry = new Entry().setResource(condition);
bundle.addEntry(conditionsEntry);
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryDstu2Bundle.java, line 158 (Poor Error Handling: Overly Broad Catch) — Low

```
}
} catch (Exception e) {
logger.error("Error in getting Condition Data");
}

```

sof/service/LoadingQueryDstu2Bundle.java, line 180 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:180
Taint Flags:

```
Entry observationsEntry = new Entry().setResource(observation);
bundle.addEntry(observationsEntry);
}
} catch (Exception e) {
logger.error("Error in getting Observation Data");
}

```

sof/service/LoadingQueryDstu2Bundle.java, line 196 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:196
Taint Flags:

```
Entry observationsEntry = new Entry().setResource(observation);
bundle.addEntry(observationsEntry);
}
} catch (Exception e) {
logger.error("Error in getting Pregnancy Observation Data");
}
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryDstu2Bundle.java, line 196 (Poor Error Handling: Overly Broad Catch) — Low

```
199
```

sof/service/LoadingQueryDstu2Bundle.java, line 212 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:212
Taint Flags:

```
209  Entry observationsEntry = new Entry().setResource(observation);
210  bundle.addEntry(observationsEntry);
211  }
212  } catch (Exception e) {
213  logger.error("Error in getting Travel Observation Data");
214  }
215
```

sof/service/LoadingQueryDstu2Bundle.java, line 268 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:268
Taint Flags:

```
265  Entry medAdministrationEntry = new Entry().setResource(medAdministration);
266  bundle.addEntry(medAdministrationEntry);
267  }
268  } catch (Exception e) {
269  logger.error("Error in getting the MedicationAdministration Data");
270  }
271
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryDstu2Bundle.java, line 283 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:283
Taint Flags:

```
bundle.addEntry(medStatementEntry);
}
dstu2FhirData.setMedications(medStatementsList);
} catch (Exception e) {
logger.error("Error in getting the MedicationStatement Data");
}

```

sof/service/LoadingQueryDstu2Bundle.java, line 307 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:307
Taint Flags:

```
Entry diagnosticOrderEntry = new Entry().setResource(diagnosticOrder);
bundle.addEntry(diagnosticOrderEntry);
}
} catch (Exception e) {
logger.error("Error in getting the DiagnosticOrder Data");
}

```

sof/service/LoadingQueryDstu2Bundle.java, line 327 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryDstu2Bundle.java, line 327 (Poor Error Handling: Overly Broad Catch) — Low

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:327
Taint Flags:

```
Entry immunizationEntry = new Entry().setResource(immunization);
bundle.addEntry(immunizationEntry);
}
} catch (Exception e) {
logger.error("Error in getting the Immunization Data");
}

```

sof/service/LoadingQueryDstu2Bundle.java, line 347 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:347
Taint Flags:

```
Entry diagnosticReportEntry = new Entry().setResource(diagnosticReport);
bundle.addEntry(diagnosticReportEntry);
}
} catch (Exception e) {
logger.error("Error in getting the DiagnosticReport Data");
}

```

Package: com.drajer.sof.utils

sof/utils/RefreshTokenScheduler.java, line 126 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.utils

sof/utils/RefreshTokenScheduler.java, line 126 (Poor Error Handling: Overly Broad Catch) — Low

Enclosing Method: updateAccessToken()
File: sof/utils/RefreshTokenScheduler.java:126
Taint Flags:

```
existingAuthDetails.setLastUpdated(new Date());
authDetailsService.saveOrUpdate(existingAuthDetails);
logger.info("Successfully updated AccessToken value in database");
} catch (Exception e) {
logger.error("Error in Updating the AccessToken value into database: " + e.getMessage());
}

```

sof/utils/Authorization.java, line 95 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getAccessToken()
File: sof/utils/Authorization.java:95
Taint Flags:

```
logger.info("Received AccessToken for Client: {}", tokenDetails.getClientId());
logger.info("Received AccessToken: {}", tokenResponse);

} catch (Exception e) {
logger.error(
"Error in Getting the AccessToken for the client: " + tokenDetails.getClientId());
}
```

sof/utils/RefreshTokenScheduler.java, line 110 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getAccessToken()
File: sof/utils/RefreshTokenScheduler.java:110
Taint Flags:

```
logger.info("Received AccessToken: {}", tokenResponse);
updateAccessToken(authDetails, tokenResponse);
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.utils

sof/utils/RefreshTokenScheduler.java, line 110 (Poor Error Handling: Overly Broad Catch) — Low

```
} catch (Exception e) {
logger.error("Error in Getting the AccessToken for the client: " +
authDetails.getClientId());
}
return tokenResponse;
```

sof/utils/FhirContextInitializer.java, line 69 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: submitResource()
File: sof/utils/FhirContextInitializer.java:69
Taint Flags:

```
MethodOutcome outcome = new MethodOutcome();
try {
outcome = genericClient.create().resource(resource).prettyPrint().encodedJson().execute();
} catch (Exception e) {
logger.error("Error in Submitting the resource::::" + resource.getResourceType().name());
}

```

sof/utils/FhirContextInitializer.java, line 88 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getResouceById()
File: sof/utils/FhirContextInitializer.java:88
Taint Flags:

```
resource = genericClient.read().resource(resourceName).withId(resourceId).execute();
// logger.info(resourceName + ":::::::::::::::::" +
// context.newJsonParser().encodeResourceToString(resource));
} catch (Exception e) {
logger.error("Error in getting " + resourceName + " resource by Id: " + resourceId, e);
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.utils

sof/utils/FhirContextInitializer.java, line 88 (Poor Error Handling: Overly Broad Catch) — Low

```
}
return resource;
```

sof/utils/Authorization.java, line 42 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getMetadata()
File: sof/utils/Authorization.java:42
Taint Flags:

```
39  response = restTemplate.exchange(serverURL, HttpMethod.GET, entity, String.class);
40  metadata = new JSONObject(response.getBody());
41  logger.info("Received Metadata Information from URL:::: {}", serverURL);
42  } catch (Exception e) {
43  logger.error("Error in getting Metadata information for URL::::" + serverURL);
44  }
45  return metadata;
```

sof/utils/R4ResourcesData.java, line 855 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:855
Taint Flags:

```
852  BundleEntryComponent patientEntry = new BundleEntryComponent();
853  patientEntry.setResource(patient);
854  bundle.addEntry(patientEntry);
855  } catch (Exception e) {
856  logger.error("Error in getting Patient Data");
857  }
858  // Step 1: Get Encounters for Patient based on encId. (Create a method to get
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.utils

sof/utils/R4ResourcesData.java, line 942 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:942
Taint Flags:

```
}
BundleEntryComponent encounterEntry = new BundleEntryComponent().setResource(encounter);
bundle.addEntry(encounterEntry);
} catch (Exception e) {
logger.error("Error in getting Encounter Data");
}

```

sof/utils/R4ResourcesData.java, line 966 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:966
Taint Flags:

```
BundleEntryComponent conditionsEntry = new BundleEntryComponent().setResource(condition);
bundle.addEntry(conditionsEntry);
}
} catch (Exception e) {
logger.error("Error in getting Condition Data");
}

```

sof/utils/R4ResourcesData.java, line 990 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.utils

sof/utils/R4ResourcesData.java, line 990 (Poor Error Handling: Overly Broad Catch) — Low

Sink Details

Sink: CatchBlock
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:990
Taint Flags:

```
new BundleEntryComponent().setResource(observation);
bundle.addEntry(observationsEntry);
}
} catch (Exception e) {
logger.error("Error in getting Observation Data");
}

```

sof/utils/R4ResourcesData.java, line 1048 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:1048
Taint Flags:

```
new BundleEntryComponent().setResource(medAdministration);
bundle.addEntry(medAdministrationEntry);
}
} catch (Exception e) {
logger.error("Error in getting the MedicationAdministration Data", e);
}

```

sof/utils/R4ResourcesData.java, line 1091 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:1091

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.utils

sof/utils/R4ResourcesData.java, line 1091 (Poor Error Handling: Overly Broad Catch) — Low

Taint Flags:

```
BundleEntryComponent medRequestEntry = new
BundleEntryComponent().setResource(medRequest);
bundle.addEntry(medRequestEntry);
}
} catch (Exception e) {
logger.error("Error in getting the MedicationRequest Data", e);
}

```

sof/utils/R4ResourcesData.java, line 1117 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:1117
Taint Flags:

```
new BundleEntryComponent().setResource(serviceRequest);
bundle.addEntry(serviceRequestEntry);
}
} catch (Exception e) {
logger.error("Error in getting the ServiceRequest Data");
}
return bundle;
```

sof/utils/R4ResourcesData.java, line 1132 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getResourceFromBundle()
File: sof/utils/R4ResourcesData.java:1132
Taint Flags:

```
}
}
```

Poor Error Handling: Overly Broad Catch — Low

Package: com.drajer.sof.utils

sof/utils/R4ResourcesData.java, line 1132 (Poor Error Handling: Overly Broad Catch) — Low

```
}
} catch (Exception e) {
logger.error("Error in getting the Resource from Bundle");
}
return null;
```

sof/utils/RefreshTokenScheduler.java, line 159 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getSystemAccessToken()
File: sof/utils/RefreshTokenScheduler.java:159
Taint Flags:

```
logger.info("Received AccessToken for Client: " + clientDetails.getClientId());
logger.info("Received AccessToken: {}", tokenResponse);

} catch (Exception e) {
logger.error(
"Error in Getting the AccessToken for the client: " + clientDetails.getClientId());
}
```

sof/utils/FhirContextInitializer.java, line 188 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: getResourceBundleByUrl()
File: sof/utils/FhirContextInitializer.java:188
Taint Flags:

```
bundle.getEntry().size());
}
}
} catch (Exception e) {
logger.info(
"Error in getting "
```

Poor Error Handling: Overly Broad Catch	Low
Package: com.drajer.sof.utils	
sof/utils/FhirContextInitializer.java, line 188 (Poor Error Handling: Overly Broad Catch)	**Low**

```
191   + resourceName
```

Poor Error Handling: Overly Broad Throws (6 issues)

Abstract

The method throws a generic exception making it harder for callers to do a good job of error handling and recovery.

Explanation

Declaring a method to throw `Exception` or `Throwable` makes it difficult for callers to do good error handling and error recovery. Java's exception mechanism is set up to make it easy for callers to anticipate what can go wrong and write code to handle each specific exceptional circumstance. Declaring that a method throws a generic form of exception defeats this system. **Example:** The following method throws three types of exceptions.

```
public void doExchange()
  throws IOException, InvocationTargetException,
         SQLException {
  ...
}
```

While it might seem tidier to write

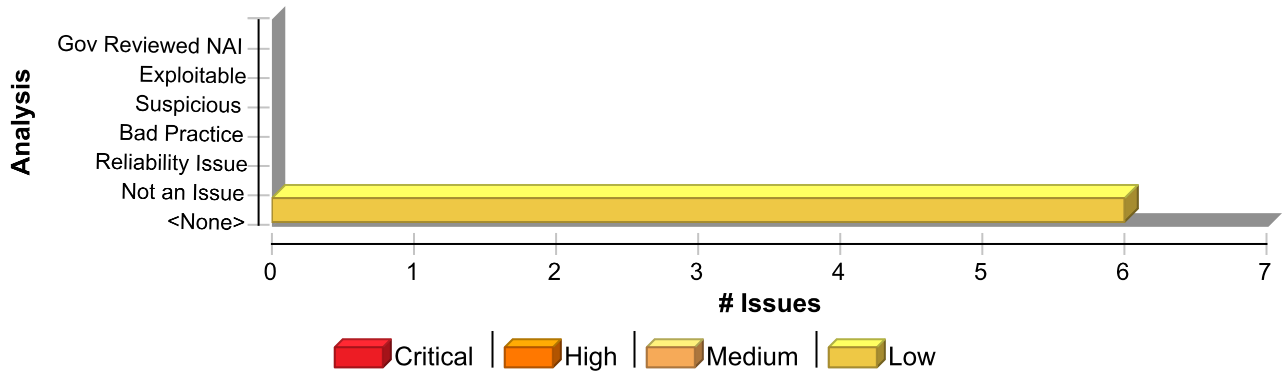
```
public void doExchange()
  throws Exception {
  ...
}
```

doing so hampers the caller's ability to understand and handle the exceptions that occur. Further, if a later revision of `doExchange()` introduces a new type of exception that should be treated differently than previous exceptions, there is no easy way to enforce this requirement.

Recommendation

Do not declare methods to throw `Exception` or `Throwable`. If the exceptions thrown by a method are not recoverable or should not generally be caught by the caller, consider throwing unchecked exceptions rather than checked exceptions. This can be accomplished by implementing exception classes that extend `RuntimeException` or `Error` instead of `Exception`, or add a try/catch wrapper in your method to convert checked exceptions to unchecked exceptions.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Overly Broad Throws	6	0	0	6
Total	**6**	**0**	**0**	**6**

Poor Error Handling: Overly Broad Throws — Low

Package: com.drajer.ecrapp.config

ecrapp/config/WebSecurityConfig.java, line 24 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: configure
Enclosing Method: configure()
File: ecrapp/config/WebSecurityConfig.java:24
Taint Flags:

```
21  private String tokenFilterClassName;
22
23  @Override
24  public void configure(WebSecurity web) throws Exception {
25  web.ignoring().antMatchers("/meta/**");
26  }
27
```

ecrapp/config/WebSecurityConfig.java, line 29 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: configure
Enclosing Method: configure()
File: ecrapp/config/WebSecurityConfig.java:29
Taint Flags:

```
26  }
27
28  @Override
29  protected void configure(HttpSecurity http) throws Exception {
30  logger.info("*********************************************************************");
31  logger.info("Security Configuration" + tokenFilterClassName);
32  logger.info("*********************************************************************");
```

Poor Error Handling: Overly Broad Throws — Low

Package: com.drajer.routing.impl

routing/impl/DirectEicrSender.java, line 72 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: sendMail
Enclosing Method: sendMail()
File: routing/impl/DirectEicrSender.java:72
Taint Flags:

```
}
}

public void sendMail(
String host,
String username,
String password,
```

routing/impl/DirectResponseReceiver.java, line 132 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: deleteMail
Enclosing Method: deleteMail()
File: routing/impl/DirectResponseReceiver.java:132
Taint Flags:

```
}
}

public void deleteMail(String host, String username, String password) throws Exception {

Properties props = new Properties();
Session session = Session.getInstance(props, null);
```

Package: com.drajer.sof.launch

sof/launch/LaunchController.java, line 259 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors

Poor Error Handling: Overly Broad Throws — Low

Package: com.drajer.sof.launch

sof/launch/LaunchController.java, line 259 (Poor Error Handling: Overly Broad Throws) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: Function: launchApp
Enclosing Method: launchApp()
File: sof/launch/LaunchController.java:259
Taint Flags:

```
256
257  @CrossOrigin
258  @RequestMapping(value = "/api/launch")
259  public void launchApp(
260  @RequestParam String launch,
261  @RequestParam String iss,
262  HttpServletRequest request,
```

sof/launch/LaunchController.java, line 333 (Poor Error Handling: Overly Broad Throws) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: redirectEndPoint
Enclosing Method: redirectEndPoint()
File: sof/launch/LaunchController.java:333
Taint Flags:

```
330
331  @CrossOrigin
332  @RequestMapping(value = "/api/redirect")
333  public void redirectEndPoint(
334  @RequestParam String code,
335  @RequestParam String state,
336  HttpServletRequest request,
```

Redundant Null Check (8 issues)

Abstract

The program can potentially dereference a null pointer, thereby causing a null pointer exception.

Explanation

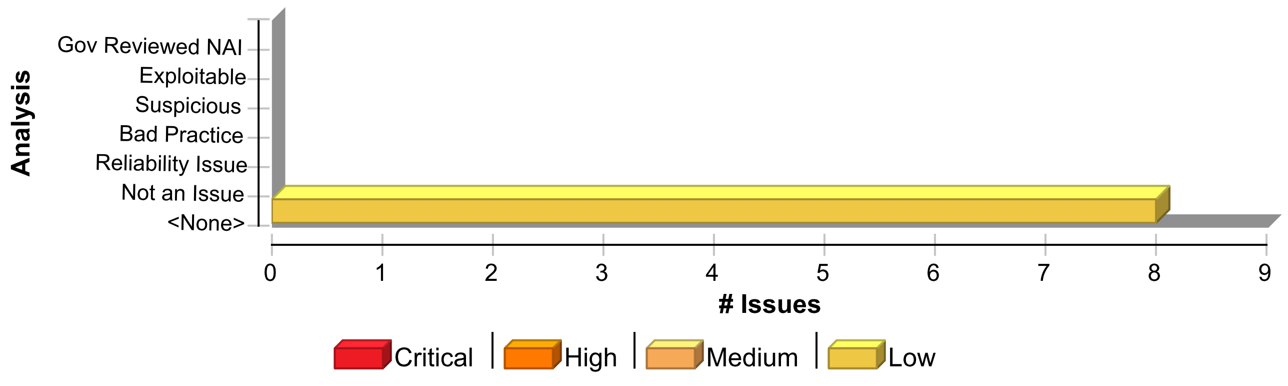
Null pointer exceptions usually occur when one or more of the programmer's assumptions is violated. A check-after-dereference error occurs when a program dereferences an object that can be `null` before checking if the object is `null`. Most null pointer issues result in general software reliability problems, but if attackers can intentionally trigger a null pointer dereference, they can use the resulting exception to bypass security logic or to cause the application to reveal debugging information that will be valuable in planning subsequent attacks. **Example:** In the following code, the programmer assumes that the variable `foo` is not `null` and confirms this assumption by dereferencing the object. However, the programmer later contradicts the assumption by checking `foo` against `null`. If `foo` can be `null` when it is checked in the `if` statement then it can also be `null` when it is dereferenced and might cause a null pointer exception. Either the dereference is unsafe or the subsequent check is unnecessary.

```
foo.setBar(val);
...
if (foo != null) {
    ...
}
```

Recommendation

Implement careful checks before dereferencing objects that might be `null`. When possible, abstract null checks into wrappers around code that manipulates resources to ensure that they are applied in all cases and to minimize the places where mistakes can occur.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Redundant Null Check	8	0	0	8
Total	**8**	**0**	**0**	**8**

Redundant Null Check — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryDstu2Bundle.java, line 255 (Redundant Null Check) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: Dereferenced : medication
Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:255
Taint Flags:

```
252 Medication medication =
253 dstu2ResourcesData.getMedicationData(
254 context, client, launchDetails, dstu2FhirData, medReference);
255 Entry medicationEntry = new Entry().setResource(medication);
256 bundle.addEntry(medicationEntry);
257 if (medication != null) {
258 List<Medication> medicationList = new ArrayList<Medication>();
```

sof/service/TriggerQueryDstu2Bundle.java, line 221 (Redundant Null Check) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: Dereferenced : medication
Enclosing Method: createDSTU2Bundle()
File: sof/service/TriggerQueryDstu2Bundle.java:221
Taint Flags:

```
218 Medication medication =
219 dstu2ResourcesData.getMedicationData(
220 context, client, launchDetails, dstu2FhirData, medReference);
221 Entry medicationEntry = new Entry().setResource(medication);
222 bundle.addEntry(medicationEntry);
223 if (medication != null) {
224 List<Medication> medicationList = new ArrayList<Medication>();
```

sof/service/LoadingQueryDstu2Bundle.java, line 255 (Redundant Null Check) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: Dereferenced : medication

Redundant Null Check — Low

Package: com.drajer.sof.service

sof/service/LoadingQueryDstu2Bundle.java, line 255 (Redundant Null Check) — Low

Enclosing Method: createDSTU2Bundle()
File: sof/service/LoadingQueryDstu2Bundle.java:255
Taint Flags:

```
Medication medication =
dstu2ResourcesData.getMedicationData(
context, client, launchDetails, dstu2FhirData, medReference);
Entry medicationEntry = new Entry().setResource(medication);
bundle.addEntry(medicationEntry);
if (medication != null) {
List<Medication> medicationList = new ArrayList<Medication>();
```

sof/service/TriggerQueryDstu2Bundle.java, line 221 (Redundant Null Check) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: Dereferenced : medication
Enclosing Method: createDSTU2Bundle()
File: sof/service/TriggerQueryDstu2Bundle.java:221
Taint Flags:

```
Medication medication =
dstu2ResourcesData.getMedicationData(
context, client, launchDetails, dstu2FhirData, medReference);
Entry medicationEntry = new Entry().setResource(medication);
bundle.addEntry(medicationEntry);
if (medication != null) {
List<Medication> medicationList = new ArrayList<Medication>();
```

Package: com.drajer.sof.utils

sof/utils/R4ResourcesData.java, line 1034 (Redundant Null Check) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: Dereferenced : medication
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:1034
Taint Flags:

```
Medication medication =
getMedicationData(context, client, launchDetails, r4FhirData, medReference);
```

Redundant Null Check — Low

Package: com.drajer.sof.utils

sof/utils/R4ResourcesData.java, line 1034 (Redundant Null Check) — Low

```
BundleEntryComponent medicationEntry =
new BundleEntryComponent().setResource(medication);
bundle.addEntry(medicationEntry);
if (medication != null) {
List<Medication> medicationList = new ArrayList<>();
```

sof/utils/R4ResourcesData.java, line 1078 (Redundant Null Check) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: Dereferenced : medication
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:1078
Taint Flags:

```
Medication medication =
getMedicationData(context, client, launchDetails, r4FhirData, medReference);
BundleEntryComponent medicationEntry =
new BundleEntryComponent().setResource(medication);
bundle.addEntry(medicationEntry);
if (medication != null) {
List<Medication> medicationList = new ArrayList<Medication>();
```

sof/utils/R4ResourcesData.java, line 1034 (Redundant Null Check) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: Dereferenced : medication
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:1034
Taint Flags:

```
Medication medication =
getMedicationData(context, client, launchDetails, r4FhirData, medReference);
BundleEntryComponent medicationEntry =
new BundleEntryComponent().setResource(medication);
bundle.addEntry(medicationEntry);
if (medication != null) {
List<Medication> medicationList = new ArrayList<>();
```

Redundant Null Check — Low

Package: com.drajer.sof.utils

sof/utils/R4ResourcesData.java, line 1078 (Redundant Null Check) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Control Flow)

Sink Details

Sink: Dereferenced : medication
Enclosing Method: getCommonResources()
File: sof/utils/R4ResourcesData.java:1078
Taint Flags:

```
1075  Medication medication =
1076  getMedicationData(context, client, launchDetails, r4FhirData, medReference);
1077  BundleEntryComponent medicationEntry =
1078  new BundleEntryComponent().setResource(medication);
1079  bundle.addEntry(medicationEntry);
1080  if (medication != null) {
1081  List<Medication> medicationList = new ArrayList<Medication>();
```

System Information Leak (2 issues)

Abstract

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation

An information leak occurs when system data or debugging information leaves the program through an output stream or logging function. **Example 1:** The following code prints an exception to the standard error stream:

```
try {
    ...
} catch (Exception e) {
    e.printStackTrace();
}
```

Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a remote user. For example, with scripting mechanisms it is trivial to redirect output information from "Standard error" or "Standard output" into a file or another program. Alternatively the system that the program runs on could have a remote logging mechanism such as a "syslog" server that will send the logs to a remote device. During development you will have no way of knowing where this information may end up being displayed. In some cases the error message tells the attacker precisely what sort of an attack the system is vulnerable to. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In the example above, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program. Here is another scenario, specific to the mobile world. Most mobile devices now implement a Near-Field Communication (NFC) protocol for quickly sharing information between devices using radio communication. It works by bringing devices to close proximity or simply having them touch each other. Even though the communication range of NFC is limited to just a few centimeters, eavesdropping, data modification and various other types of attacks are possible, since NFC alone does not ensure secure communication. **Example 2:** The Android platform provides support for NFC. The following code creates a message that gets pushed to the other device within the range.

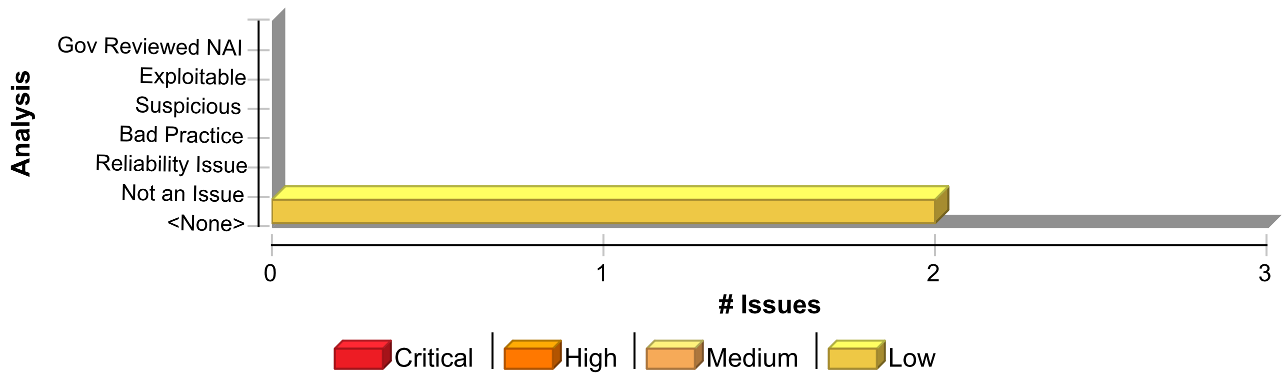
```
...
public static final String TAG = "NfcActivity";
private static final String DATA_SPLITTER = "__:DATA:__";
private static final String MIME_TYPE = "application/my.applications.mimetype";
...
public NdefMessage createNdefMessage(NfcEvent event) {
    TelephonyManager tm =
(TelephonyManager)Context.getSystemService(Context.TELEPHONY_SERVICE);
    String VERSION = tm.getDeviceSoftwareVersion();
    String text = TAG + DATA_SPLITTER + VERSION;
    NdefRecord record = new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
            MIME_TYPE.getBytes(), new byte[0], text.getBytes());
    NdefRecord[] records = { record };
    NdefMessage msg = new NdefMessage(records);
    return msg;
}
...
```

NFC Data Exchange Format (NDEF) message contains typed data, a URI, or a custom application payload. If the message contains information about the application, such as its name, MIME type, or device software version, this information could be leaked to an eavesdropper. In the example above, Fortify Static Code Analyzer reports a System Information Leak vulnerability on the return statement.

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Be careful, debugging traces can sometimes appear in non-obvious places (embedded in comments in the HTML for an error page, for example). Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system. If you are concerned about leaking system data via NFC on an Android device, you could do one of the following three things. Either do not include system data in the messages pushed to other devices in range, or encrypt the payload of the message, or establish secure communication channel at a higher layer.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak	2	0	0	2
Total	**2**	**0**	**0**	**2**

System Information Leak — Low

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 36 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: setKey()
File: ecrapp/security/AESEncryption.java:36
Taint Flags:

```
33  } catch (NoSuchAlgorithmException e) {
34  e.printStackTrace();
35  } catch (UnsupportedEncodingException e) {
36  e.printStackTrace();
37  }
38  }
```

System Information Leak — Low

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 36 (System Information Leak) — Low

```
39
```

ecrapp/security/AESEncryption.java, line 34 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: setKey()
File: ecrapp/security/AESEncryption.java:34
Taint Flags:

```
31  key = Arrays.copyOf(key, 16);
32  secretKey = new SecretKeySpec(key, "AES");
33  } catch (NoSuchAlgorithmException e) {
34  e.printStackTrace();
35  } catch (UnsupportedEncodingException e) {
36  e.printStackTrace();
37  }
```

System Information Leak: Internal (2 issues)

Abstract

Revealing system data or debugging information helps an adversary learn about the system and form a plan of attack.

Explanation

An internal information leak occurs when system data or debugging information is sent to a local file, console, or screen via printing or logging. **Example 1:** The following code prints an exception to the standard error stream:

```
try {
    ...
} catch (Exception e) {
    e.printStackTrace();
}
```

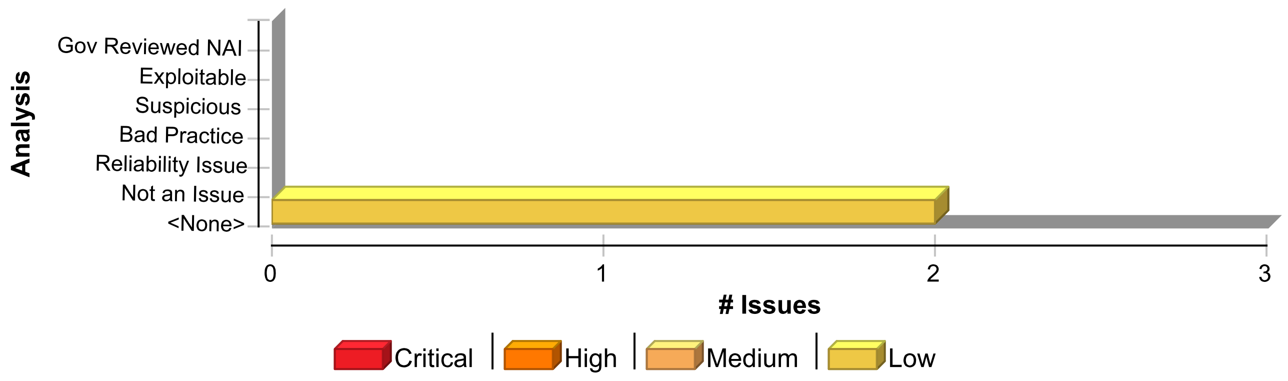
Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a user. In some cases the error message tells the attacker precisely what sort of an attack the system is vulnerable to. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In the example above, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program. In the mobile world, information leaks are also a concern. **Example 2:** The code below logs the stack trace of a caught exception on the Android platform.

```
...
try {
  ...
} catch (Exception e) {
    Log.e(TAG, Log.getStackTraceString(e));
}
...
```

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Be careful, debugging traces can sometimes appear in non-obvious places (embedded in comments in the HTML for an error page, for example). Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak: Internal	2	0	0	2
Total	**2**	**0**	**0**	**2**

System Information Leak: Internal — Low

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 59 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read e
From: com.drajer.ecrapp.security.AESEncryption.decrypt
File: ecrapp/security/AESEncryption.java:59

```
56  cipher.init(Cipher.DECRYPT_MODE, secretKey);
57  return new
String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
58  } catch (Exception e) {
59  System.out.println("Error while decrypting: " + e.toString());
60  }
61  return null;
62  }
```

Sink Details

Sink: java.io.PrintStream.println()
Enclosing Method: decrypt()
File: ecrapp/security/AESEncryption.java:59
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
56  cipher.init(Cipher.DECRYPT_MODE, secretKey);
57  return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
58  } catch (Exception e) {
59  System.out.println("Error while decrypting: " + e.toString());
```

System Information Leak: Internal — Low

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 59 (System Information Leak: Internal) — Low

```
60  }
61  return null;
62  }
```

ecrapp/security/AESEncryption.java, line 47 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: Read e
From: com.drajer.ecrapp.security.AESEncryption.encrypt
File: ecrapp/security/AESEncryption.java:47

```
44  cipher.init(Cipher.ENCRYPT_MODE, secretKey);
45  return
Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8'
46  } catch (Exception e) {
47  System.out.println("Error while encrypting: " + e.toString());
48  }
49  return null;
50  }
```

Sink Details

Sink: java.io.PrintStream.println()
Enclosing Method: encrypt()
File: ecrapp/security/AESEncryption.java:47
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
44  cipher.init(Cipher.ENCRYPT_MODE, secretKey);
45  return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
46  } catch (Exception e) {
47  System.out.println("Error while encrypting: " + e.toString());
48  }
49  return null;
50  }
```

Weak Cryptographic Hash (1 issue)

Abstract

Weak cryptographic hashes cannot guarantee data integrity and should not be used in security-critical contexts.

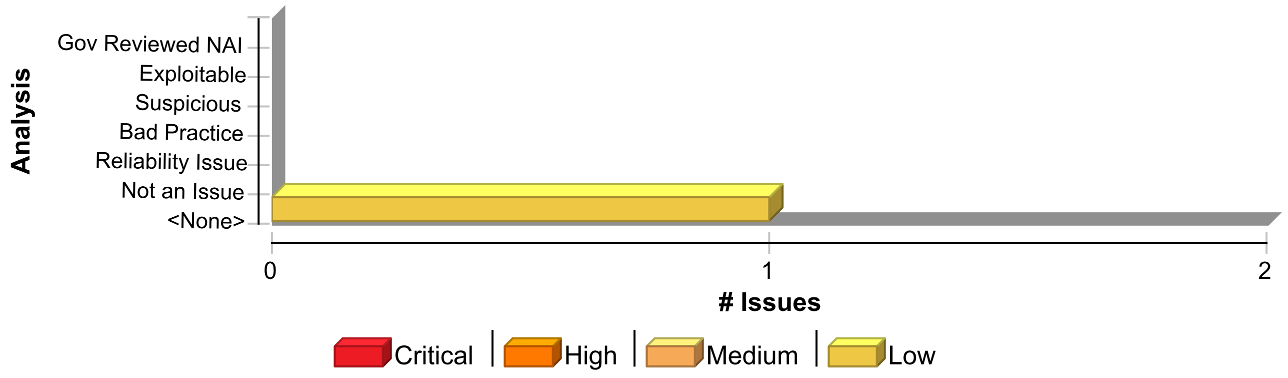
Explanation

MD2, MD4, MD5, RIPEMD-160, and SHA-1 are popular cryptographic hash algorithms often used to verify the integrity of messages and other data. However, as recent cryptanalysis research has revealed fundamental weaknesses in these algorithms, they should no longer be used within security-critical contexts. Effective techniques for breaking MD and RIPEMD hashes are widely available, so those algorithms should not be relied upon for security. In the case of SHA-1, current techniques still require a significant amount of computational power and are more difficult to implement. However, attackers have found the Achilles' heel for the algorithm, and techniques for breaking it will likely lead to the discovery of even faster attacks.

Recommendation

Discontinue the use of MD2, MD4, MD5, RIPEMD-160, and SHA-1 for data-verification in security-critical contexts. Currently, SHA-224, SHA-256, SHA-384, SHA-512, and SHA-3 are good alternatives. However, these variants of the Secure Hash Algorithm have not been scrutinized as closely as SHA-1, so be mindful of future research that might impact the security of these algorithms.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Weak Cryptographic Hash	1	0	0	1
Total	**1**	**0**	**0**	**1**

Weak Cryptographic Hash **Low**

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 29 (Weak Cryptographic Hash) **Low**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Weak Cryptographic Hash	Low

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 29 (Weak Cryptographic Hash)	Low

Sink Details

Sink: getInstance()
Enclosing Method: setKey()
File: ecrapp/security/AESEncryption.java:29
Taint Flags:

```
26  MessageDigest sha = null;
27  try {
28  key = myKey.getBytes("UTF-8");
29  sha = MessageDigest.getInstance("SHA-1");
30  key = sha.digest(key);
31  key = Arrays.copyOf(key, 16);
32  secretKey = new SecretKeySpec(key, "AES");
```

Weak Encryption: Insecure Mode of Operation (2 issues)

Abstract

Cryptographic encryption algorithms should not be used with an insecure mode of operation.

Explanation

A mode of operation of a block cipher is an algorithm that describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block. Some of the modes of operation include Electronic Codebook (ECB), Cipher Block Chaining (CBC), and Cipher Feedback (CFB). ECB mode is inherently weak, as it results in the same ciphertext for identical blocks of plain text. CBC mode is the superior choice as it does not have this weakness. **Example 1:** The following code uses AES cipher with ECB mode:

```
...
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher cipher = Cipher.getInstance("AES/ECB/PKCS7Padding", "BC");
cipher.init(Cipher.ENCRYPT_MODE, key);
...
```

Cipher Transformation Modes: The first argument to `Cipher.getInstance` is a string parameter `transformation` in the form "algorithm/mode/padding" or "algorithm". If the mode is not specified, then the mode selected is the provider-specific default, which is likely Electronic Codebook (ECB) mode for Java and Android. ECB mode is inherently a weaker encryption mode because identical blocks of plain text is encrypted into identical blocks of ciphertext. CBC (cipher-block chaining) mode is superior because it does not have this weakness. **Example**: gaining a Cipher instance with the weak ECB transformation mode:

```
Cipher c = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

Example: gaining a Cipher instance with default transformation mode, which could be the weak ECB mode:

```
Cipher c = Cipher.getInstance("AES");
```

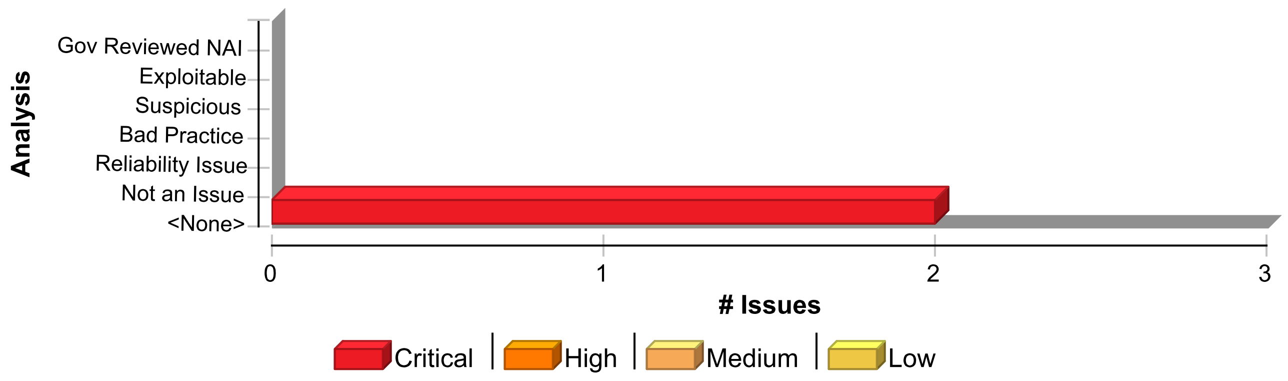
This finding is from research found in "An Empirical Study of Cryptographic Misuse in Android Applications". http://www.cs.ucsb.edu/~chris/research/doc/ccs13_cryptolint.pdf

Recommendation

Avoid using ECB mode of operation when encrypting data larger than a block. CBC mode is superior as it does not produce identical blocks of ciphertext for identical blocks of plain text. However, CBC mode is somewhat inefficient and poses serious risk if used with SSL. [1] Instead, use CCM (Counter with CBC-MAC) mode, or, if performance is a concern, GCM (Galois/Counter Mode) mode where they are available. **Example 2:** The following code uses the AES cipher with CBC mode:

```
...
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding", "BC");
cipher.init(Cipher.ENCRYPT_MODE, key);
...
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Weak Encryption: Insecure Mode of Operation	2	0	0	2
Total	**2**	**0**	**0**	**2**

Weak Encryption: Insecure Mode of Operation — Critical

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 43 (Weak Encryption: Insecure Mode of Operation) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: getInstance()
Enclosing Method: encrypt()
File: ecrapp/security/AESEncryption.java:43
Taint Flags:

```
40 public static String encrypt(String strToEncrypt) {
41 try {
42 setKey(secret);
43 Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
44 cipher.init(Cipher.ENCRYPT_MODE, secretKey);
45 return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes("UTF-8")));
46 } catch (Exception e) {
```

ecrapp/security/AESEncryption.java, line 55 (Weak Encryption: Insecure Mode of Operation) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: getInstance()
Enclosing Method: decrypt()

Weak Encryption: Insecure Mode of Operation — Critical

Package: com.drajer.ecrapp.security

ecrapp/security/AESEncryption.java, line 55 (Weak Encryption: Insecure Mode of Operation) — Critical

File: ecrapp/security/AESEncryption.java:55
Taint Flags:

```
52  public static String decrypt(String strToDecrypt) {
53  try {
54  setKey(secret);
55  Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
56  cipher.init(Cipher.DECRYPT_MODE, secretKey);
57  return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
58  } catch (Exception e) {
```

About HPE Security Enterprise Security Products

HPE Security is a leading provider of security and compliance solutions for the modern enterprise that wants to mitigate risk in their hybrid environment and defend against advanced threats. Based on market-leading products from HPE Security ArcSight and HPE Security Fortify, the HPE Security Security Intelligence Platform uniquely delivers the advanced correlation, application protection, and network defenses to protect today's hybrid IT infrastructure from sophisticated cyber threats.