Fortify Standalone Report Generator

Developer Workbook

health-apis-patient-generated-data-fortify-202101252214

Table of Contents

Executive Summary
Project Description
Issue Breakdown by Fortify Categories
Results Outline

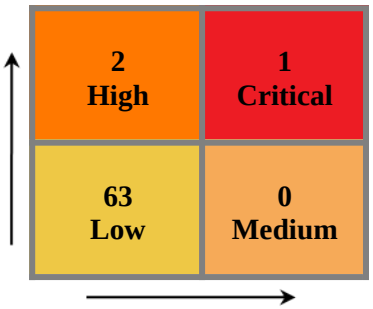
Executive Summary

This workbook is intended to provide all necessary details and information for a developer to understand and remediate the different issues discovered during the health-apis-patient-generated-data-fortify-202101252214 project audit. The information contained in this workbook is targeted at project managers and developers.

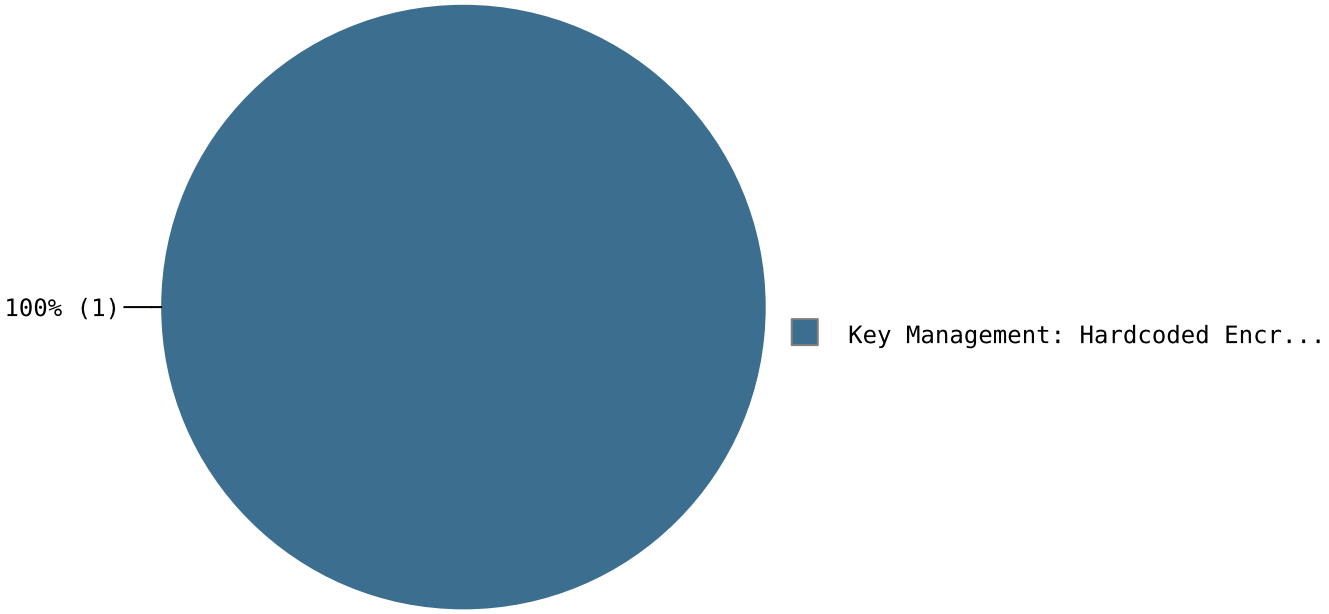
This section provides an overview of the issues uncovered during analysis.

Project Name:	health-apis-patient-generate data-fortify-202101252214
Project Version:	
SCA:	Results Present
WebInspect:	Results Not Present
WebInspect Agent:	Results Not Present
Other:	Results Not Present

Issues by Priority



Top Ten Critical Categories



Project Description

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

SCA

Date of Last Analysis:	Jan 25, 2021, 10:15 PM	**Engine Version:**	20.2.1.0010
Host Name:	29c2005ff16e	**Certification:**	VALID
Number of Files:	47	**Lines of Code:**	1,447

Rulepack Name	Rulepack Version
Fortify Secure Coding Rules, Extended, JSP	2020.4.0.0007
Fortify Secure Coding Rules, Core, Java	2020.4.0.0007
Fortify Secure Coding Rules, Extended, Content	2020.4.0.0007
Fortify Secure Coding Rules, Core, Android	2020.4.0.0007
Fortify Secure Coding Rules, Core, Annotations	2020.4.0.0007
Fortify Secure Coding Rules, Extended, Java	2020.4.0.0007
Fortify Secure Coding Rules, Extended, Configuration	2020.4.0.0007

Issue Breakdown by Fortify Categories

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

Category	Fortify Priority (audited/total)				Total Issues
	Critical	High	Medium	Low	
Build Misconfiguration: External Maven Dependency Repository	0	0	0	0 / 2	0 / 2
Code Correctness: Class Does Not Implement equals	0	0	0	0 / 3	0 / 3
Dead Code: Expression is Always false	0	0	0	0 / 1	0 / 1
Dead Code: Unused Method	0	0	0	0 / 2	0 / 2
J2EE Bad Practices: Leftover Debug Code	0	0	0	0 / 1	0 / 1
Key Management: Hardcoded Encryption Key	0 / 1	0	0	0	0 / 1
Missing Check for Null Parameter	0	0	0	0 / 9	0 / 9
Password Management: Password in Configuration File	0	0 / 1	0	0	0 / 1
Poor Error Handling: Overly Broad Catch	0	0	0	0 / 2	0 / 2
Poor Style: Confusing Naming	0	0	0	0 / 41	0 / 41
Portability Flaw: Locale Dependent Comparison	0	0 / 1	0	0	0 / 1
Redundant Null Check	0	0	0	0 / 2	0 / 2

Results Outline

Build Misconfiguration: External Maven Dependency Repository (2 issues)

Abstract

This maven build script relies on external sources, which could allow an attacker to insert malicious code into the final product or to take control of the build machine.

Explanation

Several tools exist within the Java development world to aid in dependency management: both Apache Ant and Apache Maven build systems include functionality specifically designed to help manage dependencies and Apache Ivy is developed explicitly as a dependency manager. Although there are differences in their behavior, these tools share the common functionality that they automatically download external dependencies specified in the build process at build time. This makes it much easier for developer B to build software in the same manner as developer A. Developers just store dependency information in the build file, which means that each developer and build engineer has a consistent way to obtain dependencies, compile the code, and deploy without the dependency management hassles involved in manual dependency management. The following examples illustrate how Ivy, Ant, and Maven can be used to manage external dependencies as part of a build process. Under Maven, instead of listing explicit URLs from which to retrieve the dependencies, developers specify the dependency names and versions and Maven relies on its underlying configuration to identify the server(s) from which to retrieve the dependencies. For commonly used components this saves the developer from having to researching dependency locations. **Example 1:** The following except from a Maven pom.xml file shows how a developer can specify multiple external dependencies using their name and version:

```
<dependencies>
  <dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.1</version>
  </dependency>
  <dependency>
    <groupId>javax.jms</groupId>
    <artifactId>jms</artifactId>
    <version>1.1</version>
  </dependency>
  ...
</dependencies>
```

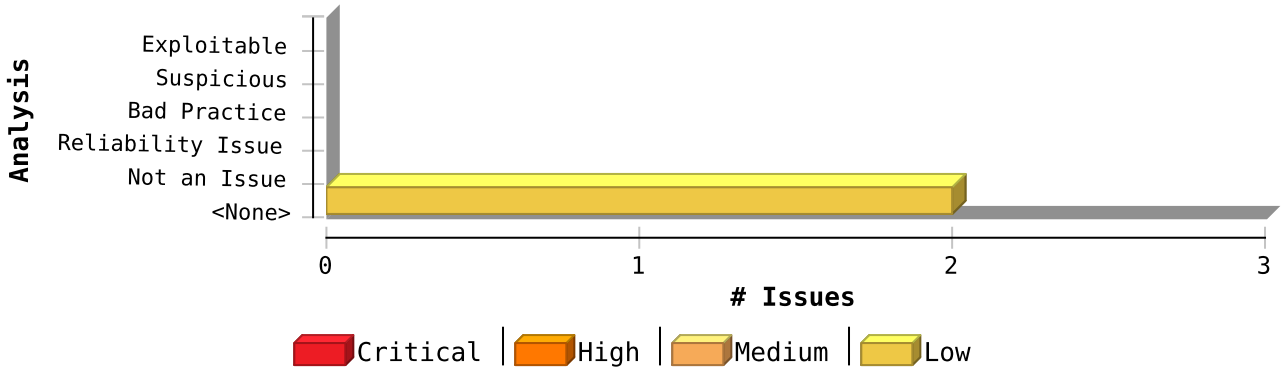
Two distinct types of attack scenarios affect these systems: An attacker could either compromise the server hosting the dependency or compromise the DNS server the build machine uses to redirect requests for hostname of the server hosting the dependency to a machine controlled by the attacker. Both scenarios result in the attacker gaining the ability to inject a malicious version of a dependency into a build running on an otherwise uncompromised machine. Regardless of the attack vector used to deliver the Trojan dependency, these scenarios share the common element that the build system blindly accepts the malicious binary and includes it in the build. Because the build system has no recourse for rejecting the malicious binary and existing security mechanisms, such as code review, typically focus on internally-developed code rather than external dependencies, this type of attack has a strong potential to go unnoticed as it spreads through the development environment and potentially into production. Although there is some risk of a compromised dependency being introduced into a manual build process, by the tendency of automated build systems to retrieve the dependency from an external source each time the build system is run in a new environment greatly increases the window of opportunity for an attacker. An attacker need only compromise the dependency server or the DNS server during one of the many times the dependency is retrieved in order to compromise the machine on which the build is occurring.

Recommendation

The simplest solution is to refrain from adopting automated dependency management systems altogether. Managing dependencies manually eliminates the potential for unexpected behavior caused by the build system. Obviously, an attacker could still mount one of the attacks described previously to coincide with the manual retrieval of a dependency, but limiting the frequency with which the dependency must be retrieved significantly reduces the window of opportunity for an attacker. Finally, this solution forces the development organization to rely on what is ostensibly an antiquated build system. A system based on manual dependency management is often more difficult to use and maintain, and might be unacceptable in some software development environments. The second solution is a hybrid of the traditional manual dependency management approach and the fully automated solution that is popular today. The biggest advantage of the manual build process is the decreased window of attack, which can be achieved in a semi-automated system by replicating external dependency servers internally. Any build system that requires an external dependency can then point to the internal server using a hard-coded internal IP address to bypass the risk of DNS-based attacks. As new dependencies are added and new versions released, they can be downloaded once and included on the internal repository. This solution reduces the attack opportunities and allows the organization leverage existing internal network security infrastructure. To implement this solution using Maven, a project should have the IP address for an internal repository hard coded the pom.xml. Specifying the IP address in the pom.xml ensures the internal repository will be used by the corresponding build, but is tied to a specific project. Alternatively, the IP address can be specified in settings.xml, which makes the configuration easier to share across multiple projects.
Example 2: The following Maven pom.xml demonstrates the use of an explicit internal IP address (the entries can also be used in settings.xml):

```
<project>
  ...
  <repositories>
    <repository>
      <releases>
        <enabled>true</enabled>
        <updatePolicy>always</updatePolicy>
        <checksumPolicy>warn</checksumPolicy>
      </releases>
      <snapshots>
        <enabled>true</enabled>
        <updatePolicy>never</updatePolicy>
        <checksumPolicy>fail</checksumPolicy>
      </snapshots>
      <id>central</id>
      <name>Internal Repository</name>
      <url>http://172.16.1.13/maven2</url>
      <layout>default</layout>
    </repository>
  </repositories>
  <pluginRepositories>
    ...
  </pluginRepositories>
  ...
</project>
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Build Misconfiguration: External Maven Dependency Repository	2	0	0	2
Total	**2**	**0**	**0**	**2**

Build Misconfiguration: External Maven Dependency Repository — Low

Package: <none>

pom.xml, line 2 (Build Misconfiguration: External Maven Dependency Repository) — Low

Issue Details

Kingdom: Environment
Scan Engine: SCA (Configuration)

Sink Details

Sink: //project/repositories
File: pom.xml:2
Taint Flags:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3  <modelVersion>4.0.0</modelVersion>
4  <parent>
5  <groupId>gov.va.api.health</groupId>
6
7 undefined
```

Package: patient-generated-data

patient-generated-data/pom.xml, line 2 (Build Misconfiguration: External Maven Dependency Repository) — Low

Issue Details

Kingdom: Environment
Scan Engine: SCA (Configuration)

Sink Details

Build Misconfiguration: External Maven Dependency Repository	Low

Package: patient-generated-data

patient-generated-data/pom.xml, line 2 (Build Misconfiguration: External Maven Dependency Repository)	Low

Sink: //project/repositories
File: patient-generated-data/pom.xml:2
Taint Flags:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3  <modelVersion>4.0.0</modelVersion>
4  <parent>
5  <groupId>gov.va.api.health</groupId>
6
7 undefined
```

Code Correctness: Class Does Not Implement equals (3 issues)

Abstract

The `equals()` method is called on an object that does not implement `equals()`.

Explanation

When comparing objects, developers usually want to compare properties of objects. However, calling `equals()` on a class (or any super class/interface) that does not explicitly implement `equals()` results in a call to the `equals()` method inherited from `java.lang.Object`. Instead of comparing object member fields or other properties, `Object.equals()` compares two object instances to see if they are the same. Although there are legitimate uses of `Object.equals()`, it is often an indication of buggy code. **Example 1:**

```
public class AccountGroup
{
    private int gid;

    public int getGid()
    {
        return gid;
    }

    public void setGid(int newGid)
    {
        gid = newGid;
    }
}
...
public class CompareGroup
{
    public boolean compareGroups(AccountGroup group1, AccountGroup group2)
    {
        return group1.equals(group2);   //equals() is not implemented in
AccountGroup
    }
}
```

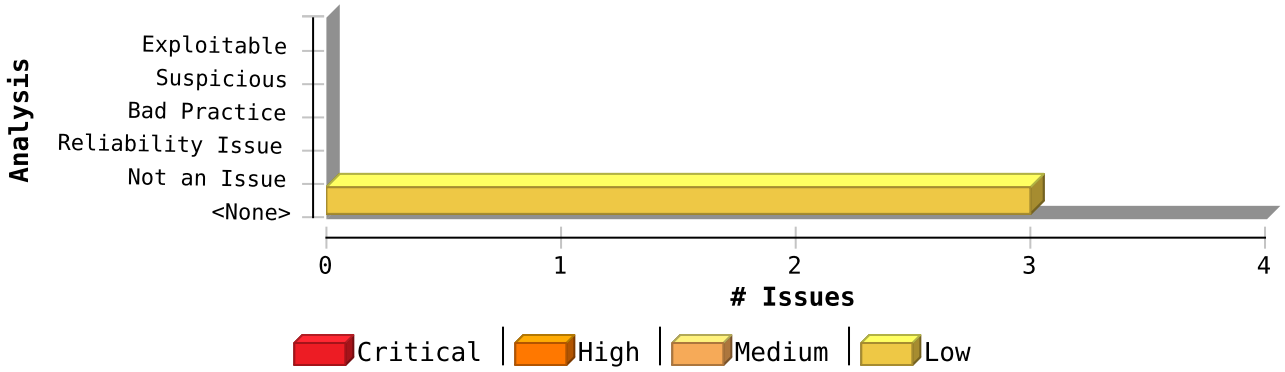
Recommendation

Verify that the use of `Object.equals()` is really the method you intend to call. If not, implement an `equals()` method or use a different method for comparing objects. **Example 2:** The following code adds an `equals()` method to the example from the Explanation section.

```
public class AccountGroup
{
    private int gid;

    public int getGid()
    {
        return gid;
    }

    public void setGid(int newGid)
    {
        gid = newGid;
    }
```

```
    public boolean equals(Object o)
    {
        if (!(o instanceof AccountGroup))
            return false;
        AccountGroup other = (AccountGroup) o;
        return (gid == other.getGid());
    }
}
...
public class CompareGroup
{
    public static boolean compareGroups(AccountGroup group1, AccountGroup
group2)
    {
        return group1.equals(group2);
    }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Class Does Not Implement equals	3	0	0	3
Total	**3**	**0**	**0**	**3**

Code Correctness: Class Does Not Implement equals	**Low**
Package: gov.va.api.health.patientgenerateddata	
patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 78 (Code Correctness: Class Does Not Implement equals)	**Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java:78

Code Correctness: Class Does Not Implement equals — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 78 (Code Correctness: Class Does Not Implement equals) — Low

Taint Flags:

```
75 return links.isEmpty() ? null : links;
76 }
77
78 @Value
79 @Builder
80 public static final class Bundling<
81 ResourceT extends Resource,
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 78 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java:78
Taint Flags:

```
75 return links.isEmpty() ? null : links;
76 }
77
78 @Value
79 @Builder
80 public static final class Bundling<
81 ResourceT extends Resource,
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 78 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java:78
Taint Flags:

```
75 return links.isEmpty() ? null : links;
76 }
```

Code Correctness: Class Does Not Implement equals	Low
Package: gov.va.api.health.patientgenerateddata	
patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 78 (Code Correctness: Class Does Not Implement equals)	**Low**

```
77
78 @Value
79 @Builder
80 public static final class Bundling<
81 ResourceT extends Resource,
```

Dead Code: Expression is Always false (1 issue)

Abstract

This expression will always evaluate to `false`.

Explanation

This expression will always evaluate to `false`; the program could be rewritten in a simpler form. The nearby code may be present for debugging purposes, or it may not have been maintained along with the rest of the program. The expression may also be indicative of a bug earlier in the method. **Example 1:** The following method never sets the variable `secondCall` after initializing it to `false`. (The variable `firstCall` is mistakenly used twice.) The result is that the expression `firstCall && secondCall` will always evaluate to `false`, so `setUpDualCall()` will never be invoked.

```
public void setUpCalls() {
  boolean firstCall = false;
  boolean secondCall = false;

  if (fCall > 0) {
    setUpFCall();
    firstCall = true;
  }
  if (sCall > 0) {
    setUpSCall();
    firstCall = true;
  }

  if (firstCall && secondCall) {
    setUpDualCall();
  }
}
```

Example 2: The following method never sets the variable `firstCall` to `true`. (The variable `firstCall` is mistakenly set to `false` after the first conditional statement.) The result is that the first part of the expression `firstCall && secondCall` will always evaluate to `false`.

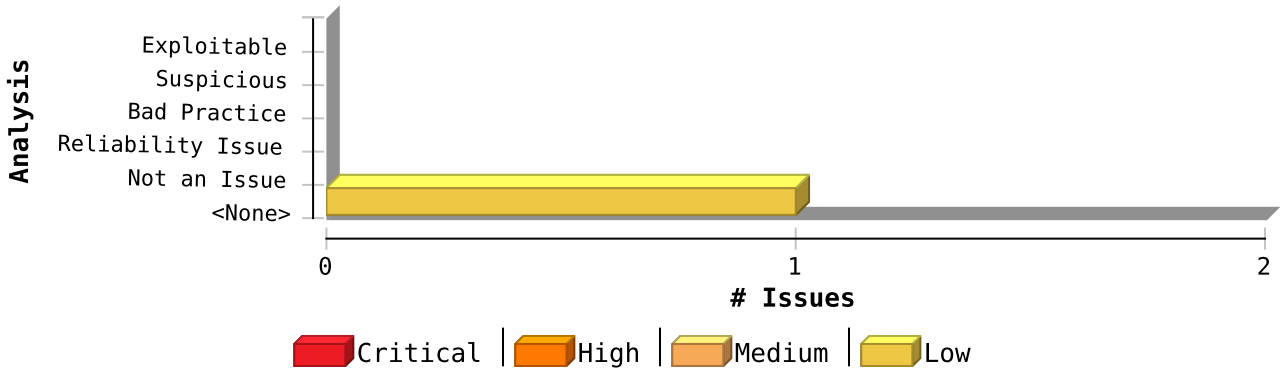
```
public void setUpCalls() {
  boolean firstCall = false;
  boolean secondCall = false;

  if (fCall > 0) {
    setUpFCall();
    firstCall = false;
  }
  if (sCall > 0) {
    setUpSCall();
    secondCall = true;
  }

  if (firstCall && secondCall) {
    setUpForCall();
  }
}
```

Recommendation

In general, you should repair or remove unused code. It causes additional complexity and maintenance burden without

contributing to the functionality of the program.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dead Code: Expression is Always false	1	0	0	1
Total	**1**	**0**	**0**	**1**

Dead Code: Expression is Always false — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/Controllers.java, line 15 (Dead Code: Expression is Always false) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: checkRequestState()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/Controllers.java:15
Taint Flags:

```
12 /** Wrapper for Preconditions.checkState which throws a BadRequest. */
13 @SneakyThrows
14 public static void checkRequestState(boolean condition, @NonNull String message) {
15 try {
16 checkState(condition, message);
17 } catch (IllegalStateException e) {
18 throw new Exceptions.BadRequest(e.getMessage(), e);
```

Dead Code: Unused Method (2 issues)

Abstract

This method is not reachable from any method outside the class.

Explanation

This method is never called or is only called from other dead code. **Example 1:** In the following class, the method `doWork()` can never be called.

```
public class Dead {
  private void doWork() {
    System.out.println("doing work");
  }
  public static void main(String[] args) {
    System.out.println("running Dead");
  }
}
```

Example 2: In the following class, two private methods call each other, but since neither one is ever invoked from anywhere else, they are both dead code.

```
public class DoubleDead {
  private void doTweedledee() {
    doTweedledumb();
  }
  private void doTweedledumb() {
    doTweedledee();
  }
  public static void main(String[] args) {
    System.out.println("running DoubleDead");
  }
}
```

(In this case it is a good thing that the methods are dead: invoking either one would cause an infinite loop.)

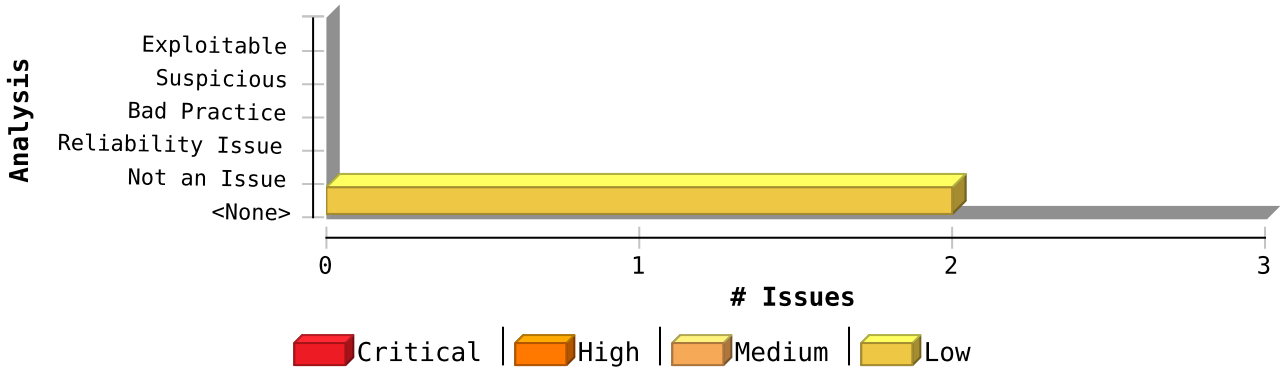
Recommendation

A dead method may indicate a bug in dispatch code. **Example 3:** If method is flagged as dead named `getWitch()` in a class that also contains the following dispatch method, it may be because of a copy-and-paste error. The 'w' case should return `getWitch()` not `getMummy()`.

```
public ScaryThing getScaryThing(char st) {
  switch(st) {
    case 'm':
      return getMummy();
    case 'w':
      return getMummy();
    default:
      return getBlob();
  }
}
```

In general, you should repair or remove dead code. To repair dead code, execute the dead code directly or indirectly through a public method. Dead code causes additional complexity and maintenance burden without contributing to the functionality of the program.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Dead Code: Unused Method	2	0	0	2
Total	**2**	**0**	**0**	**2**

Dead Code: Unused Method — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/TokenListMapping.java, line 27 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: codingJoin
Enclosing Method: codingJoin()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/TokenListMapping.java:27
Taint Flags:

```
24 return "|" + str + "|";
25 }
26
27 private static Stream<String> codingJoin(Coding tag) {
28 if (tag == null) {
29 return Stream.empty();
30 }
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/CompositeMapping.java, line 41 (Dead Code: Unused Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: valueJoin

Dead Code: Unused Method	Low
Package: gov.va.api.health.patientgenerateddata	
patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/CompositeMapping.java, line 41 (Dead Code: Unused Method)	**Low**

Enclosing Method: valueJoin()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/CompositeMapping.java:41
Taint Flags:

```
38 .collect(joining(","));
39 }
40
41 private static Stream<String> valueJoin(UsageContext context) {
42 if (context == null
43 || context.valueCodeableConcept() == null
44 || context.valueCodeableConcept().coding() == null) {
```

J2EE Bad Practices: Leftover Debug Code (1 issue)

Abstract

Debug code can create unintended entry points in a deployed web application.

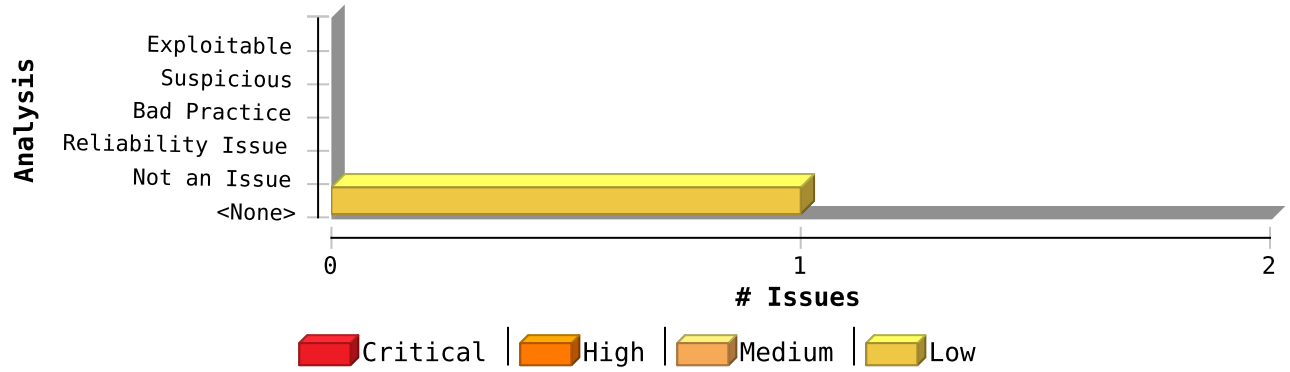
Explanation

A common development practice is to add "back door" code specifically designed for debugging or testing purposes that is not intended to be shipped or deployed with the application. When this sort of debug code is accidentally left in the application, the application is open to unintended modes of interaction. These back door entry points create security risks because they are not considered during design or testing and fall outside of the expected operating conditions of the application. The most common example of forgotten debug code is a `main()` method appearing in a web application. Although this is an acceptable practice during product development, classes that are part of a production J2EE application should not define a `main()`.

Recommendation

Remove debug code before deploying a production version of an application. Regardless of whether a direct security threat can be articulated, it is unlikely that there is a legitimate reason for such code to remain in the application after the early stages of development.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: Leftover Debug Code	1	0	0	1
Total	**1**	**0**	**0**	**1**

J2EE Bad Practices: Leftover Debug Code	Low
Package: gov.va.api.health.patientgenerateddata	
patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/Application.java, line 8 (J2EE Bad Practices: Leftover Debug Code)	**Low**

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

J2EE Bad Practices: Leftover Debug Code	Low
Package: gov.va.api.health.patientgenerateddata	
patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/Application.java, line 8 (J2EE Bad Practices: Leftover Debug Code)	**Low**

Sink: Function: main
Enclosing Method: main()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/Application.java:8
Taint Flags:

```
5
6 @SpringBootApplication
7 public class Application {
8  public static void main(String[] args) {
9  SpringApplication.run(Application.class, args);
10  }
11 }
```

Key Management: Hardcoded Encryption Key (1 issue)

Abstract

Hardcoded encryption keys can compromise security in a way that cannot be easily remedied.

Explanation

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

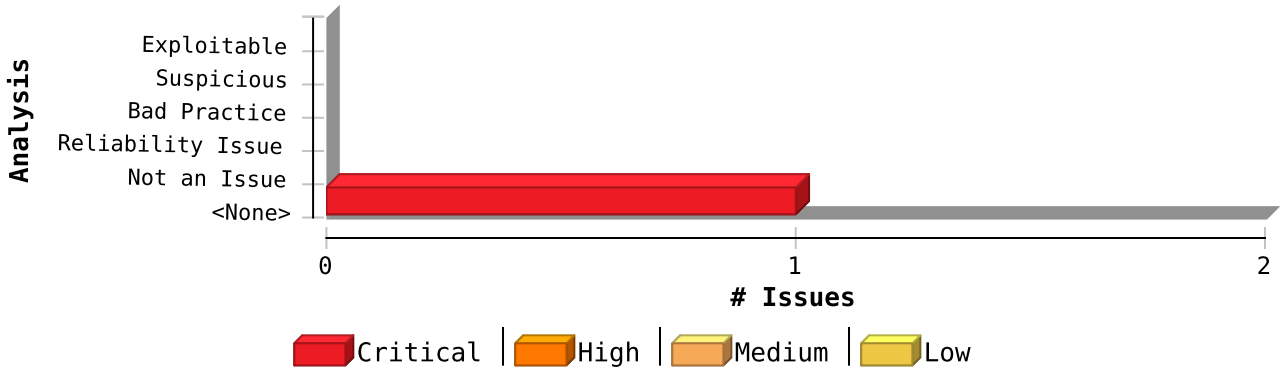
```
...
private static final String encryptionKey = "lakdsljkalkjlksdfkl";
byte[] keyBytes = encryptionKey.getBytes();
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher encryptCipher = Cipher.getInstance("AES");
encryptCipher.init(Cipher.ENCRYPT_MODE, key);
...
```

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

Recommendation

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

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Key Management: Hardcoded Encryption Key	1	0	0	1
Total	**1**	**0**	**0**	**1**

Key Management: Hardcoded Encryption Key	Critical
Package: gov.va.api.health.patientgenerateddata	
patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/WebExceptionHandler.java, line 49 (Key Management: Hardcoded Encryption Key)	**Critical**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: WebExceptionHandler()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/WebExceptionHandler.java:49
Taint Flags:

```
46 private final String encryptionKey;
47
48 public WebExceptionHandler(@Value("${web-exception-key}") String encryptionKey) {
49 checkState(!"unset".equals(encryptionKey), "web-exception-key is unset");
50 this.encryptionKey = encryptionKey;
51 }
52
```

Missing Check for Null Parameter (9 issues)

Abstract

This function violates the contract that it must compare its parameter with `null`.

Explanation

The Java standard requires that implementations of `Object.equals()`, `Comparable.compareTo()`, and `Comparator.compare()` must return a specified value if their parameters are `null`. Failing to follow this contract may result in unexpected behavior. **Example 1:** The following implementation of the `equals()` method does not compare its parameter with `null`.

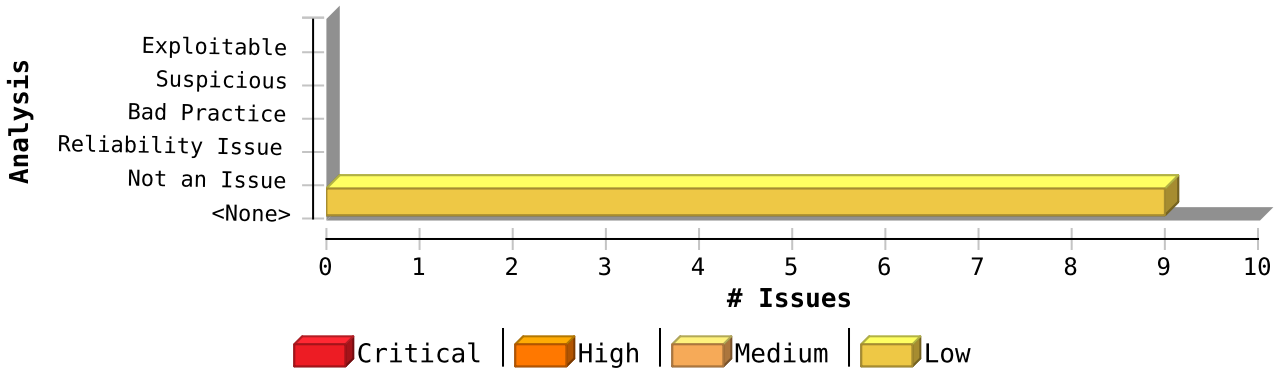
```
public boolean equals(Object object)
{
   return (toString().equals(object.toString()));
}
```

Recommendation

Always follow the contract that `Object.equals()` must return `null` if it receives a `null` parameter. **Example 2:** The previous example is rewritten to explicitly check for a `null` argument and return false if one is found.

```
public boolean equals(Object object)
{
    if (object == null)
        return false;
    return (toString().equals(object.toString()));
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Missing Check for Null Parameter	9	0	0	9
Total	**9**	**0**	**0**	**9**

Missing Check for Null Parameter — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 78 (Missing Check for Null Parameter) — Low

Issue Details

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

Sink Details

Sink: o = #param(0)
Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java:78
Taint Flags:

```
75 return links.isEmpty() ? null : links;
76 }
77
78 @Value
79 @Builder
80 public static final class Bundling<
81 ResourceT extends Resource,
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java, line 193 (Missing Check for Null Parameter) — Low

Issue Details

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

Sink Details

Sink: o = #param(0)
Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java:193
Taint Flags:

```
190 private final CapabilityStatement.SearchParamType type;
191 }
192
193 @Value
194 @Builder
195 static final class SupportedResource {
196 String type;
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/TokenListMapping.java, line 16 (Missing Check for Null Parameter) — Low

Issue Details

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

Missing Check for Null Parameter — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/TokenListMapping.java, line 16 (Missing Check for Null Parameter) — Low

Sink Details

Sink: o = #param(0)
Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/TokenListMapping.java:16
Taint Flags:

```
13 import lombok.Value;
14 import org.springframework.data.jpa.domain.Specification;
15
16 @Value
17 @Builder
18 public final class TokenListMapping<EntityT> implements SingleParameterMapping<EntityT> {
19  String parameterName;
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/CompositeMapping.java, line 20 (Missing Check for Null Parameter) — Low

Issue Details

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

Sink Details

Sink: o = #param(0)
Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/CompositeMapping.java:20
Taint Flags:

```
17 import org.springframework.data.jpa.domain.Specification;
18
19 /** See https://www.hl7.org/fhir/r4/search.html#composite */
20 @Value
21 @Builder
22 public final class CompositeMapping<EntityT> implements SingleParameterMapping<EntityT> {
23  String parameterName;
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java, line 15 (Missing Check for Null Parameter) — Low

Issue Details

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

Sink Details

Sink: o = #param(0)
Enclosing Method: equals()

Missing Check for Null Parameter — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java, line 15 (Missing Check for Null Parameter) — **Low**

File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java:15
Taint Flags:

```
12 @Configuration
13 @EnableConfigurationProperties
14 @ConfigurationProperties("metadata")
15 @Data
16 @Accessors(fluent = false)
17 @NoArgsConstructor
18 @AllArgsConstructor
```

Package: gov.va.api.health.patientgenerateddata.observation

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/observation/ObservationEntity.java, line 25 (Missing Check for Null Parameter) — **Low**

Issue Details

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

Sink Details

Sink: o = #param(0)
Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/observation/ObservationEntity.java:25
Taint Flags:

```
22 @Table(name = "Observation", schema = "app")
23 @NoArgsConstructor(access = AccessLevel.PRIVATE)
24 @AllArgsConstructor(access = AccessLevel.PRIVATE)
25 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
26 public class ObservationEntity implements PayloadEntity<Observation> {
27  @Id @EqualsAndHashCode.Include private String id;
28
```

Package: gov.va.api.health.patientgenerateddata.patient

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientEntity.java, line 25 (Missing Check for Null Parameter) — **Low**

Issue Details

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

Sink Details

Sink: o = #param(0)
Enclosing Method: equals()

Missing Check for Null Parameter — Low

Package: gov.va.api.health.patientgenerateddata.patient

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientEntity.java, line 25 (Missing Check for Null Parameter) — **Low**

File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientEntity.java:25
Taint Flags:

```
22 @Table(name = "Patient", schema = "app")
23 @NoArgsConstructor(access = AccessLevel.PRIVATE)
24 @AllArgsConstructor(access = AccessLevel.PRIVATE)
25 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
26 public class PatientEntity implements PayloadEntity<Patient> {
27  @Id @EqualsAndHashCode.Include private String id;
28
```

Package: gov.va.api.health.patientgenerateddata.questionnaire

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java, line 26 (Missing Check for Null Parameter) — **Low**

Issue Details

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

Sink Details

Sink: o = #param(0)
Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java:26
Taint Flags:

```
23 @Table(name = "Questionnaire", schema = "app")
24 @NoArgsConstructor(access = AccessLevel.PRIVATE)
25 @AllArgsConstructor(access = AccessLevel.PRIVATE)
26 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
27 public class QuestionnaireEntity implements PayloadEntity<Questionnaire> {
28  @Id @EqualsAndHashCode.Include private String id;
29
```

Package: gov.va.api.health.patientgenerateddata.questionnaireresponse

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 27 (Missing Check for Null Parameter) — **Low**

Issue Details

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

Sink Details

Sink: o = #param(0)

Missing Check for Null Parameter	Low
Package: gov.va.api.health.patientgenerateddata.questionnaireresponse	
patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 27 (Missing Check for Null Parameter)	**Low**

Enclosing Method: equals()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java:27
Taint Flags:

```
24 @Table(name = "QuestionnaireResponse", schema = "app")
25 @NoArgsConstructor(access = AccessLevel.PRIVATE)
26 @AllArgsConstructor(access = AccessLevel.PRIVATE)
27 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
28 public class QuestionnaireResponseEntity implements PayloadEntity<QuestionnaireResponse> {
29  @Id @EqualsAndHashCode.Include private String id;
30
```

Password Management: Password in Configuration File (1 issue)

Abstract

Storing a plain text password in a configuration file may result in a system compromise.

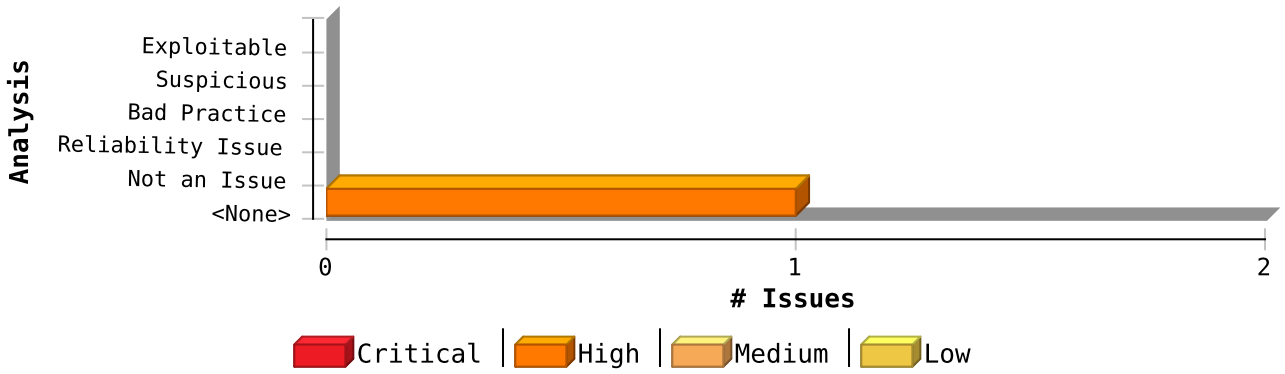
Explanation

Storing a plain text password in a configuration file allows anyone who can read the file access to the password-protected resource. Developers sometimes believe that they cannot defend the application from someone who has access to the configuration, but this attitude makes an attacker's job easier. Good password management guidelines require that a password never be stored in plain text.

Recommendation

A password should never be stored in plain text. An administrator should be required to enter the password when the system starts. If that approach is impractical, a less secure but often adequate solution is to obfuscate the password and scatter the de-obfuscation material around the system so that an attacker has to obtain and correctly combine multiple system resources to decipher the password. 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 solution the only viable option is a proprietary one.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Password Management: Password in Configuration File	1	0	0	1
Total	**1**	**0**	**0**	**1**

Password Management: Password in Configuration File	**High**
Package: .src.main.resources	
patient-generated-data/src/main/resources/application.properties, line 14 (Password Management: Password in Configuration File)	**High**

Issue Details

Kingdom: Environment
Scan Engine: SCA (Configuration)

Password Management: Password in Configuration File	High
Package: .src.main.resources	
patient-generated-data/src/main/resources/application.properties, line 14 (Password Management: Password in Configuration File)	**High**

Sink Details

Sink: spring.datasource.password
File: patient-generated-data/src/main/resources/application.properties:14
Taint Flags:

```
11 server.tomcat.relaxed-query-chars=|
12 spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver
13 spring.datasource.hikari.minimum-idle=2
14 spring.datasource.password=unset
15 spring.datasource.url=unset
16 spring.datasource.username=unset
17 spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

Poor Error Handling: Overly Broad Catch (2 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:

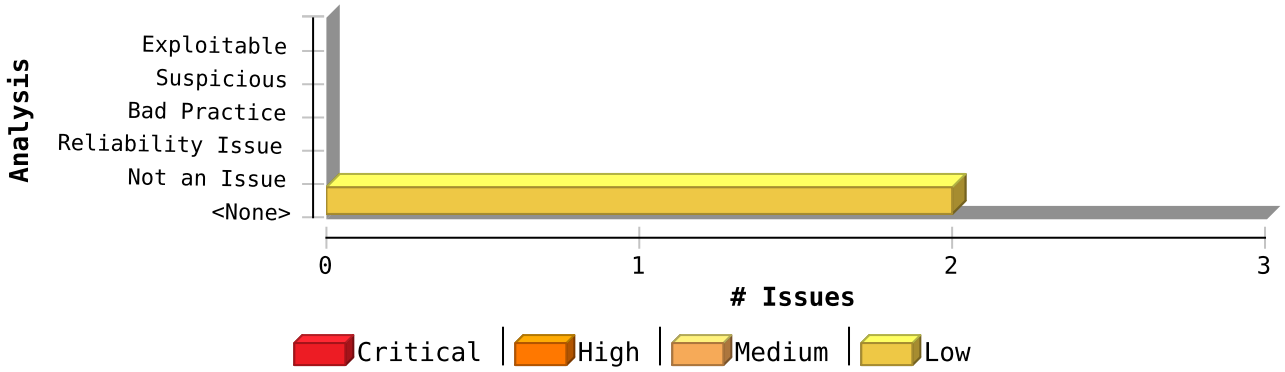
```
  try {
    doExchange();
  }
  catch (Exception e) {
    logger.error("doExchange failed", e);
  }
```

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	2	0	0	2
Total	**2**	**0**	**0**	**2**

Poor Error Handling: Overly Broad Catch	Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/PayloadEntity.java, line 19 (Poor Error Handling: Overly Broad Catch)	Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: deserializePayload()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/PayloadEntity.java:19
Taint Flags:

```
16 try {
17 checkState(payload() != null);
18 return MAPPER.readValue(payload(), resourceType());
19 } catch (Exception e) {
20 throw new Exceptions.InvalidPayload(id(), e);
21 }
22 }
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/InternalClientKeyFilterConfiguration.java, line 52 (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: gov.va.api.health.patientgenerateddata	
patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/InternalClientKeyFilterConfiguration.java, line 52 (Poor Error Handling: Overly Broad Catch)	**Low**

Sink: CatchBlock
Enclosing Method: doFilterInternal()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/InternalClientKeyFilterConfiguration.java:52
Taint Flags:

```
49 try {
50 verify(requestKey);
51 filterChain.doFilter(request, response);
52 } catch (Exception e) {
53 resolver.resolveException(request, response, null, e);
54 }
55 }
```

Poor Style: Confusing Naming (41 issues)

Abstract

The class contains a field and a method with the same name.

Explanation

It is confusing to have a member field and a method with the same name. It makes it easy for a programmer to accidentally call the method when attempting to access the field or vice versa. **Example 1:**

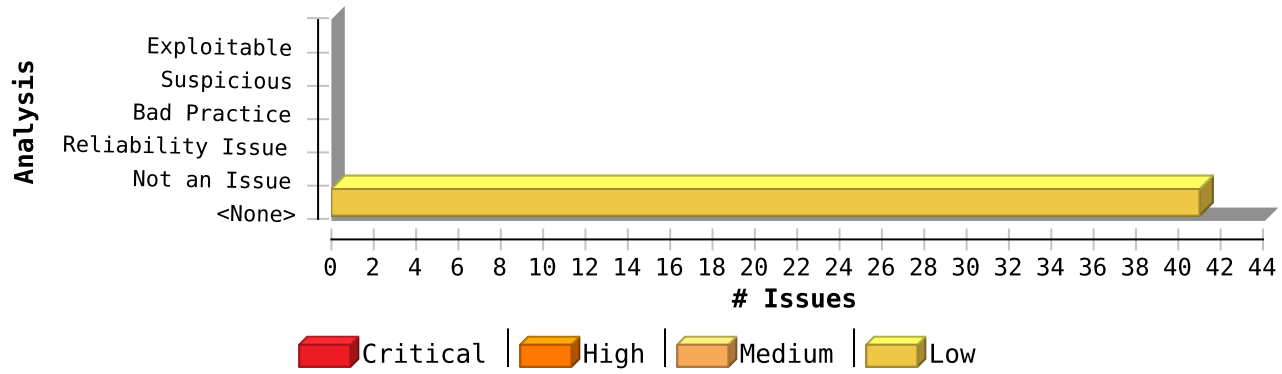
```
public class Totaller {
  private int total;
  public int total() {
    ...
  }
}
```

Recommendation

Rename either the method or the field. If the method returns the field, consider following the standard getter/setter naming convention. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
public class Totaller {
  private int total;
  public int getTotal() {
    ...
  }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Style: Confusing Naming	41	0	0	41
Total	**41**	**0**	**0**	**41**

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/TokenListMapping.java, line 21 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: fieldName
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/TokenListMapping.java:21
Taint Flags:

```
18 public final class TokenListMapping<EntityT> implements SingleParameterMapping<EntityT> {
19  String parameterName;
20
21  String fieldName;
22
23  static String addTerminators(String str) {
24  return "|" + str + "|";
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/TokenListMapping.java, line 19 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: parameterName
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/TokenListMapping.java:19
Taint Flags:

```
16 @Value
17 @Builder
18 public final class TokenListMapping<EntityT> implements SingleParameterMapping<EntityT> {
19  String parameterName;
20
21  String fieldName;
22
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java, line 21 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java, line 21 (Poor Style: Confusing Naming) — Low

Sink: Field: endpointAuthorize
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java:21
Taint Flags:

```
18 @AllArgsConstructor
19 @Builder
20 public class MetadataProperties {
21  private String endpointAuthorize;
22  private String endpointManagement;
23  private String endpointRevocation;
24  private String endpointToken;
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java, line 45 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: observationRepository
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java:45
Taint Flags:

```
42 public class ManagementController {
43  private final LinkProperties linkProperties;
44
45  @Getter private final ObservationRepository observationRepository;
46
47  @Getter private final PatientRepository patientRepository;
48
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java, line 196 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: type
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java:196
Taint Flags:

```
193  @Value
194  @Builder
```

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java, line 196 (Poor Style: Confusing Naming) — Low

```
195 static final class SupportedResource {
196 String type;
197
198 String profileUrl;
199
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/LinkProperties.java, line 21 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: maxPageSize
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/LinkProperties.java:21
Taint Flags:

```
18 public class LinkProperties {
19 private final int defaultPageSize;
20
21 private final int maxPageSize;
22
23 private final String r4Url;
24
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/LinkProperties.java, line 23 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: r4Url
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/LinkProperties.java:23
Taint Flags:

```
20
21 private final int maxPageSize;
22
23 private final String r4Url;
24
25 @Builder
26 @Autowired
```

Poor Style: Confusing Naming	Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/LinkProperties.java, line 23 (Poor Style: Confusing Naming) — **Low**

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java, line 24 (Poor Style: Confusing Naming) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: endpointToken
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java:24
Taint Flags:

```
21 private String endpointAuthorize;
22 private String endpointManagement;
23 private String endpointRevocation;
24 private String endpointToken;
25 }
26
27 undefined
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java, line 23 (Poor Style: Confusing Naming) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: endpointRevocation
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java:23
Taint Flags:

```
20 public class MetadataProperties {
21 private String endpointAuthorize;
22 private String endpointManagement;
23 private String endpointRevocation;
24 private String endpointToken;
25 }
26
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 84 (Poor Style: Confusing Naming) — **Low**

Issue Details

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 84 (Poor Style: Confusing Naming) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: newBundle
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java:84
Taint Flags:

```
81 ResourceT extends Resource,
82 EntryT extends AbstractEntry<ResourceT>,
83 BundleT extends AbstractBundle<EntryT>> {
84 @NonNull private final Supplier<BundleT> newBundle;
85
86 @NonNull private final Supplier<EntryT> newEntry;
87
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/CompositeMapping.java, line 23 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: parameterName
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/CompositeMapping.java:23
Taint Flags:

```
20 @Value
21 @Builder
22 public final class CompositeMapping<EntityT> implements SingleParameterMapping<EntityT> {
23 String parameterName;
24
25 String fieldName;
26
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java, line 47 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java, line 47 (Poor Style: Confusing Naming) — Low

Sink: Field: patientRepository
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java:47
Taint Flags:

```
44
45 @Getter private final ObservationRepository observationRepository;
46
47 @Getter private final PatientRepository patientRepository;
48
49 @Getter private final QuestionnaireRepository questionnaireRepository;
50
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java, line 200 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: searches
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java:200
Taint Flags:

```
197
198 String profileUrl;
199
200 Set<SearchParam> searches;
201
202 CapabilityStatement.CapabilityResource asResource() {
203 return CapabilityStatement.CapabilityResource.builder()
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java, line 22 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: endpointManagement
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java:22
Taint Flags:

```
19 @Builder
20 public class MetadataProperties {
```

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataProperties.java, line 22 (Poor Style: Confusing Naming) — Low

```
21 private String endpointAuthorize;
22 private String endpointManagement;
23 private String endpointRevocation;
24 private String endpointToken;
25 }
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 88 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: linkProperties
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java:88
Taint Flags:

```
85
86 @NonNull private final Supplier<EntryT> newEntry;
87
88 @NonNull private final LinkProperties linkProperties;
89
90 public static <
91 ResourceT extends Resource,
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java, line 49 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: questionnaireRepository
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java:49
Taint Flags:

```
46
47 @Getter private final PatientRepository patientRepository;
48
49 @Getter private final QuestionnaireRepository questionnaireRepository;
50
51 @Getter private final QuestionnaireResponseRepository questionnaireResponseRepository;
52
```

Poor Style: Confusing Naming	Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java, line 49 (Poor Style: Confusing Naming)	Low

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/CompositeMapping.java, line 25 (Poor Style: Confusing Naming)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: fieldName
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/CompositeMapping.java:25
Taint Flags:

```
22 public final class CompositeMapping<EntityT> implements SingleParameterMapping<EntityT> {
23  String parameterName;
24
25  String fieldName;
26
27  static String addTerminators(String str) {
28  return "|" + str + "|";
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java, line 190 (Poor Style: Confusing Naming)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: type
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java:190
Taint Flags:

```
187
188  private final String param;
189
190  private final CapabilityStatement.SearchParamType type;
191  }
192
193  @Value
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java, line 51 (Poor Style: Confusing Naming)	Low

Issue Details

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java, line 51 (Poor Style: Confusing Naming) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: questionnaireResponseRepository
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ManagementController.java:51
Taint Flags:

```
48
49 @Getter private final QuestionnaireRepository questionnaireRepository;
50
51 @Getter private final QuestionnaireResponseRepository questionnaireResponseRepository;
52
53 @PostMapping(value = "/Observation")
54 ResponseEntity<Observation> create(@Valid @RequestBody Observation observation) {
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java, line 198 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: profileUrl
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java:198
Taint Flags:

```
195 static final class SupportedResource {
196 String type;
197
198 String profileUrl;
199
200 Set<SearchParam> searches;
201
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/LinkProperties.java, line 19 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/LinkProperties.java, line 19 (Poor Style: Confusing Naming) — Low

Sink: Field: defaultPageSize
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/LinkProperties.java:19
Taint Flags:

```
16 @Getter
17 @Component
18 public class LinkProperties {
19  private final int defaultPageSize;
20
21  private final int maxPageSize;
22
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java, line 86 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: newEntry
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/VulcanizedBundler.java:86
Taint Flags:

```
83  BundleT extends AbstractBundle<EntryT>> {
84  @NonNull private final Supplier<BundleT> newBundle;
85
86  @NonNull private final Supplier<EntryT> newEntry;
87
88  @NonNull private final LinkProperties linkProperties;
89
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java, line 188 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: param
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java:188
Taint Flags:

```
185  SUBJECT("subject", CapabilityStatement.SearchParamType.reference),
186  TAG("_tag", CapabilityStatement.SearchParamType.token);
```

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/MetadataController.java, line 188 (Poor Style: Confusing Naming) — **Low**

```
187
188  private final String param;
189
190  private final CapabilityStatement.SearchParamType type;
191  }
```

Package: gov.va.api.health.patientgenerateddata.observation

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/observation/ObservationEntity.java, line 27 (Poor Style: Confusing Naming) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: id
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/observation/ObservationEntity.java:27
Taint Flags:

```
24 @AllArgsConstructor(access = AccessLevel.PRIVATE)
25 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
26 public class ObservationEntity implements PayloadEntity<Observation> {
27  @Id @EqualsAndHashCode.Include private String id;
28
29  @Lob
30  @Basic(fetch = FetchType.EAGER)
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/observation/ObservationEntity.java, line 31 (Poor Style: Confusing Naming) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: payload
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/observation/ObservationEntity.java:31
Taint Flags:

```
28
29  @Lob
30  @Basic(fetch = FetchType.EAGER)
31  private String payload;
32
33  @Version private Integer version;
```

Poor Style: Confusing Naming	Low

Package: gov.va.api.health.patientgenerateddata.observation

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/observation/ObservationEntity.java, line 31 (Poor Style: Confusing Naming)	Low

```
34
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/observation/ObservationEntity.java, line 33 (Poor Style: Confusing Naming)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: version
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/observation/ObservationEntity.java:33
Taint Flags:

```
30 @Basic(fetch = FetchType.EAGER)
31 private String payload;
32
33 @Version private Integer version;
34
35 @Override
36 public Class<Observation> resourceType() {
```

Package: gov.va.api.health.patientgenerateddata.patient

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientEntity.java, line 33 (Poor Style: Confusing Naming)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: version
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientEntity.java:33
Taint Flags:

```
30 @Basic(fetch = FetchType.EAGER)
31 private String payload;
32
33 @Version private Integer version;
34
35 @Override
36 public Class<Patient> resourceType() {
```

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata.patient

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientEntity.java, line 31 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: payload
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientEntity.java:31
Taint Flags:

```
28
29  @Lob
30  @Basic(fetch = FetchType.EAGER)
31  private String payload;
32
33  @Version private Integer version;
34
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientEntity.java, line 27 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: id
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientEntity.java:27
Taint Flags:

```
24 @AllArgsConstructor(access = AccessLevel.PRIVATE)
25 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
26 public class PatientEntity implements PayloadEntity<Patient> {
27  @Id @EqualsAndHashCode.Include private String id;
28
29  @Lob
30  @Basic(fetch = FetchType.EAGER)
```

Package: gov.va.api.health.patientgenerateddata.questionnaire

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java, line 30 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata.questionnaire

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java, line 30 (Poor Style: Confusing Naming) — Low

Sink Details

Sink: Field: version
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java:30
Taint Flags:

```
27 public class QuestionnaireEntity implements PayloadEntity<Questionnaire> {
28  @Id @EqualsAndHashCode.Include private String id;
29
30  @Version private Integer version;
31
32  @Lob
33  @Basic(fetch = FetchType.EAGER)
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java, line 38 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: contextTypeValue
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java:38
Taint Flags:

```
35
36  @Lob
37  @Basic(fetch = FetchType.EAGER)
38  private String contextTypeValue;
39
40  public static Sort naturalOrder() {
41  return Sort.by("id").ascending();
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java, line 34 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: payload
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java:34
Taint Flags:

```
31
```

Poor Style: Confusing Naming	Low

Package: gov.va.api.health.patientgenerateddata.questionnaire

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java, line 34 (Poor Style: Confusing Naming) — **Low**

```
32 @Lob
33 @Basic(fetch = FetchType.EAGER)
34 private String payload;
35
36 @Lob
37 @Basic(fetch = FetchType.EAGER)
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java, line 28 (Poor Style: Confusing Naming) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: id
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaire/QuestionnaireEntity.java:28
Taint Flags:

```
25 @AllArgsConstructor(access = AccessLevel.PRIVATE)
26 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
27 public class QuestionnaireEntity implements PayloadEntity<Questionnaire> {
28  @Id @EqualsAndHashCode.Include private String id;
29
30 @Version private Integer version;
31
```

Package: gov.va.api.health.patientgenerateddata.questionnaireresponse

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 39 (Poor Style: Confusing Naming) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: authored
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java:39
Taint Flags:

```
36
37 private String author;
38
```

Poor Style: Confusing Naming	Low

Package: gov.va.api.health.patientgenerateddata.questionnaireresponse

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 39 (Poor Style: Confusing Naming) **Low**

```
39  private Instant authored;
40
41  private String questionnaire;
42
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 31 (Poor Style: Confusing Naming) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: version
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java:31
Taint Flags:

```
28 public class QuestionnaireResponseEntity implements PayloadEntity<QuestionnaireResponse> {
29  @Id @EqualsAndHashCode.Include private String id;
30
31  @Version private Integer version;
32
33  @Lob
34  @Basic(fetch = FetchType.EAGER)
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 29 (Poor Style: Confusing Naming) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: id
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java:29
Taint Flags:

```
26 @AllArgsConstructor(access = AccessLevel.PRIVATE)
27 @EqualsAndHashCode(onlyExplicitlyIncluded = true)
28 public class QuestionnaireResponseEntity implements PayloadEntity<QuestionnaireResponse> {
29  @Id @EqualsAndHashCode.Include private String id;
```

Poor Style: Confusing Naming — Low

Package: gov.va.api.health.patientgenerateddata.questionnaireresponse

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 29 (Poor Style: Confusing Naming) — Low

```
30
31 @Version private Integer version;
32
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 43 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: subject
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java:43
Taint Flags:

```
40
41 private String questionnaire;
42
43 private String subject;
44
45 private String metaTag;
46
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 35 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: payload
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java:35
Taint Flags:

```
32
33 @Lob
34 @Basic(fetch = FetchType.EAGER)
35 private String payload;
36
```

Poor Style: Confusing Naming	Low

Package: gov.va.api.health.patientgenerateddata.questionnaireresponse

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 35 (Poor Style: Confusing Naming) **Low**

```
37  private String author;
38
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 37 (Poor Style: Confusing Naming) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: author
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java:37
Taint Flags:

```
34  @Basic(fetch = FetchType.EAGER)
35  private String payload;
36
37  private String author;
38
39  private Instant authored;
40
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 45 (Poor Style: Confusing Naming) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: metaTag
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java:45
Taint Flags:

```
42
43  private String subject;
44
45  private String metaTag;
46
47  public static Sort naturalOrder() {
```

Poor Style: Confusing Naming	Low

Package: gov.va.api.health.patientgenerateddata.questionnaireresponse

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 45 (Poor Style: Confusing Naming) **Low**

```
48  return Sort.by("id").ascending();
```

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java, line 41 (Poor Style: Confusing Naming) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: questionnaire
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/questionnaireresponse/QuestionnaireResponseEntity.java:41
Taint Flags:

```
38
39  private Instant authored;
40
41  private String questionnaire;
42
43  private String subject;
44
```

Portability Flaw: Locale Dependent Comparison (1 issue)

Abstract

Unexpected portability problems can be found when the locale is not specified.

Explanation

When comparing data that may be locale-dependent, an appropriate locale should be specified. **Example 1:** The following example tries to perform validation to determine if user input includes a

Recommendation

To prevent this from occurring, always make sure to either specify the default locale, or specify the locale with APIs that accept them such as `toUpperCase()`. **Example 2:** The following specifies the locale manually as an argument to `toUpperCase()`.

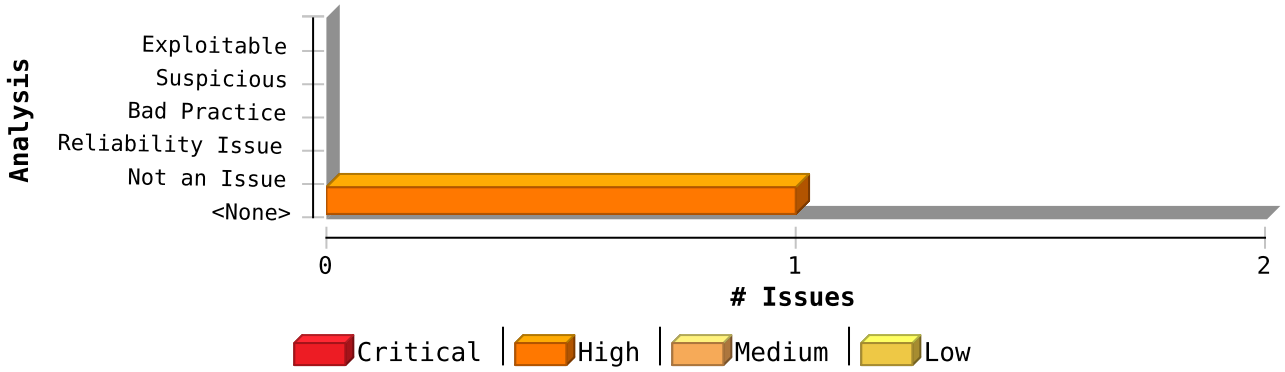
```
import java.util.Locale;
  ...
  public String tagProcessor(String tag){
    if (tag.toUpperCase(Locale.ENGLISH).equals("SCRIPT")){
      return null;
    }
    //does not contain SCRIPT tag, keep processing input
    ...
  }
  ...
```

Example 3: The following uses the function `java.lang.String.equalsIgnoreCase()` API to prevent this issue.

```
  ...
  public String tagProcessor(String tag){
    if (tag.equalsIgnoreCase("SCRIPT")){
      return null;
    }
    //does not contain SCRIPT tag, keep processing input
    ...
  }
  ...
```

This prevents the problem because `equalsIgnoreCase()` changes case similar to `Character.toLowerCase()` and `Character.toUpperCase()`. This involves creating temporary canonical forms of both strings using information from the `UnicodeData` file that is part of the Unicode Character Database maintained by the Unicode Consortium, and even though this may render them unreadable if they were to be read out, it makes comparison possible without being dependent upon locale.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Portability Flaw: Locale Dependent Comparison	1	0	0	1
Total	**1**	**0**	**0**	**1**

Portability Flaw: Locale Dependent Comparison	**High**
Package: gov.va.api.health.patientgenerateddata	
patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ ParseUtils.java, line 36 (Portability Flaw: Locale Dependent Comparison)	**High**

Issue Details

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

Sink Details

Sink: str.endsWith(...) : Comparison without checking locale
Enclosing Method: parseDateTime()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/ParseUtils.java:36
Taint Flags:

```
33 case YEAR_MONTH_DAY:
34 return parseDateTimeUtc(String.format("%sT00:00:00Z", str));
35 default:
36 if (str.endsWith("Z")) {
37 return parseDateTimeUtc(str);
38 } else {
39 return parseDateTimeOffset(str);
```

Redundant Null Check (2 issues)

Abstract

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

Explanation

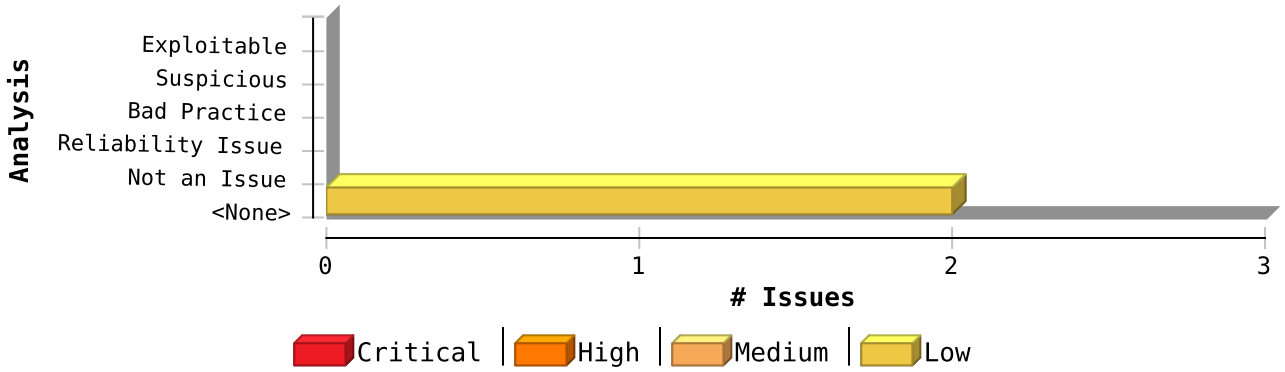
Null-pointer exceptions usually occur when one or more of the programmer's assumptions is violated. Specifically, dereference-after-check errors occur when a program makes an explicit check for `null`, but proceeds to dereference the object when it is known to be `null`. Errors of this type are often the result of a typo or programmer oversight. Most null-pointer issues result in general software reliability problems, but if attackers can intentionally cause the program to dereference a null-pointer, they can use the resulting exception to mount a denial of service attack or to cause the application to reveal debugging information that will be valuable in planning subsequent attacks. **Example 1:** In the following code, the programmer confirms that the variable `foo` is `null` and subsequently dereferences it erroneously. If `foo` is `null` when it is checked in the `if` statement, then a `null` dereference will occur, thereby causing a null-pointer exception.

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

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	2	0	0	2
Total	**2**	**0**	**0**	**2**

Redundant Null Check — Low

Package: gov.va.api.health.patientgenerateddata

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/IncludesIcnMajig.java, line 37 (Redundant Null Check) — **Low**

Issue Details

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

Sink Details

Sink: Dereferenced : serverHttpResponse
Enclosing Method: addHeader()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/IncludesIcnMajig.java:37
Taint Flags:

```
34 if (headers == null
35 || headers.get(INCLUDES_ICN_HEADER) == null
36 || headers.get(INCLUDES_ICN_HEADER).isEmpty()) {
37 serverHttpResponse.getHeaders().add(INCLUDES_ICN_HEADER, usersCsv);
38 }
39 }
40
```

Package: gov.va.api.health.patientgenerateddata.patient

patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientController.java, line 57 (Redundant Null Check) — **Low**

Issue Details

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

Sink Details

Sink: Dereferenced : patient
Enclosing Method: findIcn()
File: patient-generated-data/src/main/java/gov/va/api/health/patientgenerateddata/patient/PatientController.java:57
Taint Flags:

```
54
55 private static String findIcn(Patient patient) {
56 checkState(patient != null);
57 if (patient.identifier() != null) {
58 Optional<Identifier> mpi =
59 patient.identifier().stream()
60 .filter(
```