Fortify Standalone Report Generator

Developer Workbook

akka-persistence-query

Table of Contents

© Copyright [2008-2022] Micro Focus or one of its affiliates.

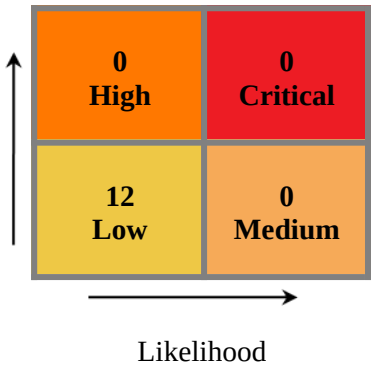
Executive Summary

This workbook is intended to provide all necessary details and information for a developer to understand and remediate the different issues discovered during the akka-persistence-query 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:	akka-persistence-query
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

This project does not contain any critical issues

© Copyright [2008-2022] Micro Focus or one of its affiliates.

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:	Jun 16, 2022, 11:35 AM	**Engine Version:**	21.1.1.0009
Host Name:	Jacks-Work-MBP.local	**Certification:**	VALID
Number of Files:	44	**Lines of Code:**	605

Rulepack Name	Rulepack Version
Fortify Secure Coding Rules, Extended, Java	2022.1.0.0007
Fortify Secure Coding Rules, Core, Scala	2022.1.0.0007
Fortify Secure Coding Rules, Extended, JSP	2022.1.0.0007
Fortify Secure Coding Rules, Core, Android	2022.1.0.0007
Fortify Secure Coding Rules, Extended, Content	2022.1.0.0007
Fortify Secure Coding Rules, Extended, Configuration	2022.1.0.0007
Fortify Secure Coding Rules, Core, Annotations	2022.1.0.0007
Fortify Secure Coding Rules, Community, Cloud	2022.1.0.0007
Fortify Secure Coding Rules, Core, Universal	2022.1.0.0007
Fortify Secure Coding Rules, Core, Java	2022.1.0.0007
Fortify Secure Coding Rules, Community, Universal	2022.1.0.0007

© Copyright [2008-2022] Micro Focus or one of its affiliates.

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	
Code Correctness: Constructor Invokes Overridable Function	0	0	0	0 / 6	0 / 6
Code Correctness: Erroneous String Compare	0	0	0	0 / 5	0 / 5
Redundant Null Check	0	0	0	0 / 1	0 / 1

© Copyright [2008-2022] Micro Focus or one of its affiliates.

Results Outline

Code Correctness: Constructor Invokes Overridable Function (6 issues)

Abstract

A constructor of the class calls a function that can be overridden.

Explanation

When a constructor calls an overridable function, it may allow an attacker to access the `this` reference prior to the object being fully initialized, which can in turn lead to a vulnerability. **Example 1:** The following calls a method that can be overridden.

```
...
class User {
  private String username;
  private boolean valid;
  public User(String username, String password){
    this.username = username;
    this.valid = validateUser(username, password);
  }
  public boolean validateUser(String username, String password){
    //validate user is real and can authenticate
    ...
  }
  public final boolean isValid(){
    return valid;
  }
}
```

Since the function `validateUser` and the class are not `final`, it means that they can be overridden, and then initializing a variable to the subclass that overrides this function would allow bypassing of the `validateUser` functionality. For example:

```
...
class Attacker extends User{
  public Attacker(String username, String password){
    super(username, password);
  }
  public boolean validateUser(String username, String password){
    return true;
  }
}
...
class MainClass{
  public static void main(String[] args){
    User hacker = new Attacker("Evil", "Hacker");
    if (hacker.isValid()){
      System.out.println("Attack successful!");
    }else{
      System.out.println("Attack failed");
    }
  }
}
```

The code in `Example 1` prints "Attack successful!", since the `Attacker` class overrides the `validateUser()` function that is called from the constructor of the superclass `User`, and Java will first look in the subclass for functions called from the constructor.

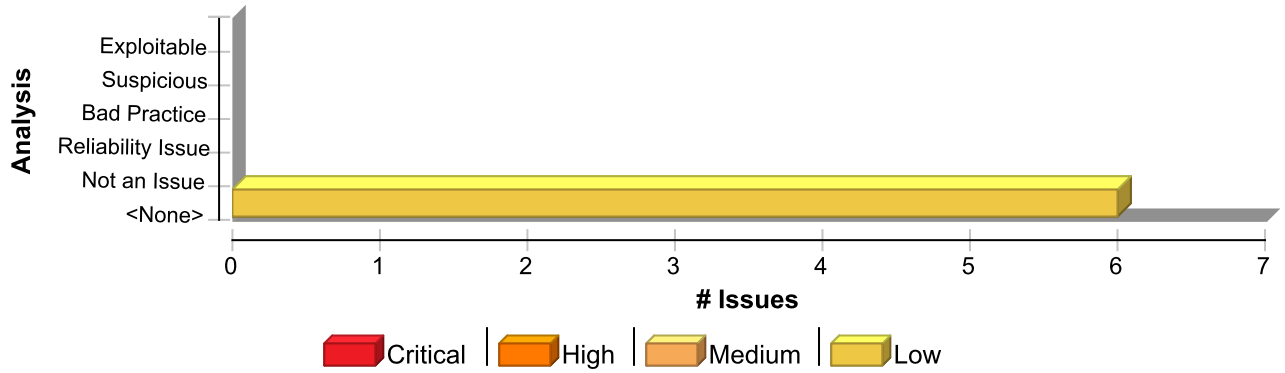
© Copyright [2008-2022] Micro Focus or one of its affiliates.

Recommendation
Constructors should not call functions that can be overridden, either by specifying them as `final`, or specifying the class as `final`. Alternatively if this code is only ever needed in the constructor, the `private` access specifier can be used, or the logic could be placed directly into the constructor of the superclass. **Example 2:** The following makes the class `final` to prevent the function from being overridden elsewhere.

```
...
final class User {
  private String username;
  private boolean valid;
  public User(String username, String password){
    this.username = username;
    this.valid = validateUser(username, password);
  }
  private boolean validateUser(String username, String password){
    //validate user is real and can authenticate
    ...
  }
  public final boolean isValid(){
    return valid;
  }
}
```

This example specifies the class as `final`, so that it cannot be subclassed, and changes the `validateUser()` function to `private`, since it is not needed elsewhere in this application. This is programming defensively, since at a later date it may be decided that the `User` class needs to be subclassed, which would result in this vulnerability reappearing if the `validateUser()` function was not set to `private`.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Constructor Invokes Overridable Function	6	0	0	6
Total	**6**	**0**	**0**	**6**

Code Correctness: Constructor Invokes Overridable Function	**Low**
Package: akka.persistence.query	
PersistenceQuery.scala, line 39 (Code Correctness: Constructor Invokes Overridable Function)	**Low**
Issue Details	

© Copyright [2008-2022] Micro Focus or one of its affiliates.

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence.query

PersistenceQuery.scala, line 39 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: pluginProvider
Enclosing Method: PersistenceQuery()
File: PersistenceQuery.scala:39
Taint Flags:

```
36 }
37
38 class PersistenceQuery(system: ExtendedActorSystem)
39  extends PersistencePlugin[scaladsl.ReadJournal, javadsl.ReadJournal, ReadJournalProvider](system)(
40  ClassTag(classOf[ReadJournalProvider]),
41  PersistenceQuery.pluginProvider)
42  with Extension {
```

Package: akka.persistence.query.journal.leveldb

journal/leveldb/LeveldbReadJournalProvider.scala, line 19 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: readJournal
Enclosing Method: LeveldbReadJournalProvider()
File: journal/leveldb/LeveldbReadJournalProvider.scala:19
Taint Flags:

```
16  override def scaladslReadJournal(): akka.persistence.query.scaladsl.ReadJournal =
17  readJournal
18
19  val javaReadJournal = new javadsl.LeveldbReadJournal(readJournal)
20
21  override def javadslReadJournal(): akka.persistence.query.javadsl.ReadJournal =
22  javaReadJournal
```

Package: akka.persistence.query.journal.leveldb.scaladsl

journal/leveldb/scaladsl/LeveldbReadJournal.scala, line 57 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

© Copyright [2008-2022] Micro Focus or one of its affiliates.

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence.query.journal.leveldb.scaladsl

journal/leveldb/scaladsl/LeveldbReadJournal.scala, line 57 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: writeJournalPluginId
Enclosing Method: LeveldbReadJournal()
File: journal/leveldb/scaladsl/LeveldbReadJournal.scala:57
Taint Flags:

```
54 private val maxBufSize: Int = config.getInt("max-buffer-size")
55
56 private val resolvedWriteJournalPluginId =
57 if (writeJournalPluginId.isEmpty)
58 system.settings.config.getString("akka.persistence.journal.plugin")
59 else
60 writeJournalPluginId
```

journal/leveldb/scaladsl/LeveldbReadJournal.scala, line 60 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: writeJournalPluginId
Enclosing Method: LeveldbReadJournal()
File: journal/leveldb/scaladsl/LeveldbReadJournal.scala:60
Taint Flags:

```
57 if (writeJournalPluginId.isEmpty)
58 system.settings.config.getString("akka.persistence.journal.plugin")
59 else
60 writeJournalPluginId
61 require(
62 resolvedWriteJournalPluginId.nonEmpty && system.settings.config
63 .getConfig(resolvedWriteJournalPluginId)
```

journal/leveldb/scaladsl/LeveldbReadJournal.scala, line 62 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

© Copyright [2008-2022] Micro Focus or one of its affiliates.

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.persistence.query.journal.leveldb.scaladsl

journal/leveldb/scaladsl/LeveldbReadJournal.scala, line 62 (Code Correctness: Constructor Invokes Overridable Function)	Low

Sink Details

Sink: FunctionCall: resolvedWriteJournalPluginId
Enclosing Method: LeveldbReadJournal()
File: journal/leveldb/scaladsl/LeveldbReadJournal.scala:62
Taint Flags:

```
59 else
60 writeJournalPluginId
61 require(
62 resolvedWriteJournalPluginId.nonEmpty && system.settings.config
63 .getConfig(resolvedWriteJournalPluginId)
64 .getString("class") == "akka.persistence.journal.leveldb.LeveldbJournal",
65 s"Leveldb read journal can only work with a Leveldb write journal. Current plugin [$resolvedWriteJournalPluginId] is not a LeveldbJournal")
```

journal/leveldb/scaladsl/LeveldbReadJournal.scala, line 64 (Code Correctness: Constructor Invokes Overridable Function)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: resolvedWriteJournalPluginId
Enclosing Method: LeveldbReadJournal()
File: journal/leveldb/scaladsl/LeveldbReadJournal.scala:64
Taint Flags:

```
61 require(
62 resolvedWriteJournalPluginId.nonEmpty && system.settings.config
63 .getConfig(resolvedWriteJournalPluginId)
64 .getString("class") == "akka.persistence.journal.leveldb.LeveldbJournal",
65 s"Leveldb read journal can only work with a Leveldb write journal. Current plugin [$resolvedWriteJournalPluginId] is not a LeveldbJournal")
66
67 /**
```


© Copyright [2008-2022] Micro Focus or one of its affiliates.

Code Correctness: Erroneous String Compare (5 issues)

Abstract

Strings should be compared with the `equals()` method, not `==` or `!=`.

Explanation

This program uses `==` or `!=` to compare two strings for equality, which compares two objects for equality, not their values. Chances are good that the two references will never be equal. **Example 1:** The following branch will never be taken.

```
  if (args[0] == STRING_CONSTANT) {
      logger.info("miracle");
  }
```

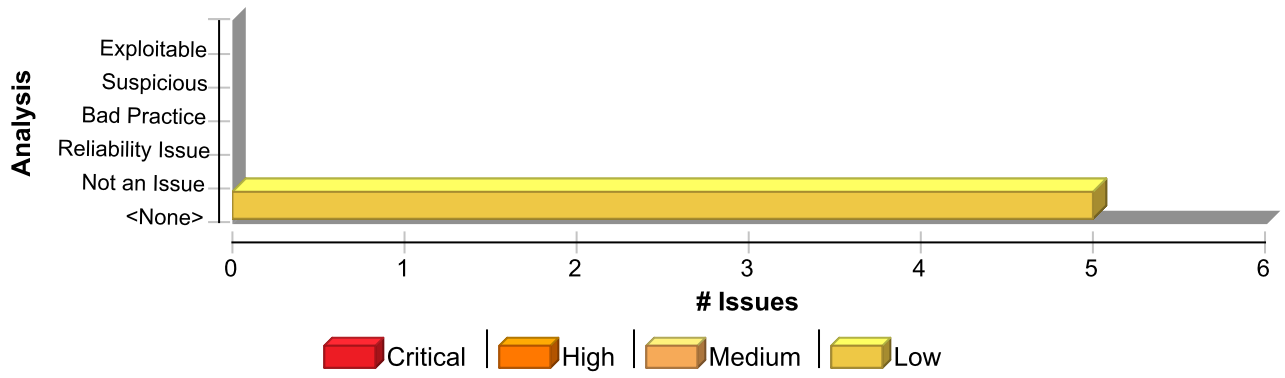
The `==` and `!=` operators will only behave as expected when they are used to compare strings contained in objects that are equal. The most common way for this to occur is for the strings to be interned, whereby the strings are added to a pool of objects maintained by the `String` class. Once a string is interned, all uses of that string will use the same object and equality operators will behave as expected. All string literals and string-valued constants are interned automatically. Other strings can be interned manually be calling `String.intern()`, which will return a canonical instance of the current string, creating one if necessary.

Recommendation

Use `equals()` to compare strings. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
  if (STRING_CONSTANT.equals(args[0])) {
      logger.info("could happen");
  }
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Erroneous String Compare	5	0	0	5
Total	**5**	**0**	**0**	**5**

© Copyright [2008-2022] Micro Focus or one of its affiliates.

Code Correctness: Erroneous String Compare — Low

Package: akka.persistence.query.internal

internal/QuerySerializer.scala, line 118 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: fromStorageRepresentation()
File: internal/QuerySerializer.scala:118
Taint Flags:

```
115 * The offset is converted from its string representation to its real type.
116 */
117 private def fromStorageRepresentation(offsetStr: String, manifest: String): Offset = {
118 manifest match {
119 case TimestampOffsetManifest => timestampOffsetFromStorageRepresentation(offsetStr)
120 case SequenceOffsetManifest => Offset.sequence(offsetStr.toLong)
121 case TimeBasedUUIDOffsetManifest => Offset.timeBasedUUID(UUID.fromString(offsetStr))
```

internal/QuerySerializer.scala, line 118 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: fromStorageRepresentation()
File: internal/QuerySerializer.scala:118
Taint Flags:

```
115 * The offset is converted from its string representation to its real type.
116 */
117 private def fromStorageRepresentation(offsetStr: String, manifest: String): Offset = {
118 manifest match {
119 case TimestampOffsetManifest => timestampOffsetFromStorageRepresentation(offsetStr)
120 case SequenceOffsetManifest => Offset.sequence(offsetStr.toLong)
121 case TimeBasedUUIDOffsetManifest => Offset.timeBasedUUID(UUID.fromString(offsetStr))
```

internal/QuerySerializer.scala, line 118 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

© Copyright [2008-2022] Micro Focus or one of its affiliates.

Code Correctness: Erroneous String Compare — Low

Package: akka.persistence.query.internal

internal/QuerySerializer.scala, line 118 (Code Correctness: Erroneous String Compare) — Low

Sink: Operation
Enclosing Method: fromStorageRepresentation()
File: internal/QuerySerializer.scala:118
Taint Flags:

```
115 * The offset is converted from its string representation to its real type.
116 */
117 private def fromStorageRepresentation(offsetStr: String, manifest: String): Offset = {
118 manifest match {
119 case TimestampOffsetManifest => timestampOffsetFromStorageRepresentation(offsetStr)
120 case SequenceOffsetManifest => Offset.sequence(offsetStr.toLong)
121 case TimeBasedUUIDOffsetManifest => Offset.timeBasedUUID(UUID.fromString(offsetStr))
```

internal/QuerySerializer.scala, line 118 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: fromStorageRepresentation()
File: internal/QuerySerializer.scala:118
Taint Flags:

```
115 * The offset is converted from its string representation to its real type.
116 */
117 private def fromStorageRepresentation(offsetStr: String, manifest: String): Offset = {
118 manifest match {
119 case TimestampOffsetManifest => timestampOffsetFromStorageRepresentation(offsetStr)
120 case SequenceOffsetManifest => Offset.sequence(offsetStr.toLong)
121 case TimeBasedUUIDOffsetManifest => Offset.timeBasedUUID(UUID.fromString(offsetStr))
```

internal/QuerySerializer.scala, line 86 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: fromBinary()
File: internal/QuerySerializer.scala:86
Taint Flags:

```
83 throw new IllegalArgumentException(s"Cannot serialize object of type [${o.getClass.getName}]")
84 }
```


© Copyright [2008-2022] Micro Focus or one of its affiliates.

Code Correctness: Erroneous String Compare	Low
Package: akka.persistence.query.internal	
internal/QuerySerializer.scala, line 86 (Code Correctness: Erroneous String Compare)	**Low**

```
85
86  override def fromBinary(bytes: Array[Byte], manifest: String): AnyRef = manifest match {
87  case EventEnvelopeManifest =>
88  val env = QueryMessages.EventEnvelope.parseFrom(bytes)
89
```


© Copyright [2008-2022] Micro Focus or one of its affiliates.

Redundant Null Check (1 issue)

Abstract

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

Explanation

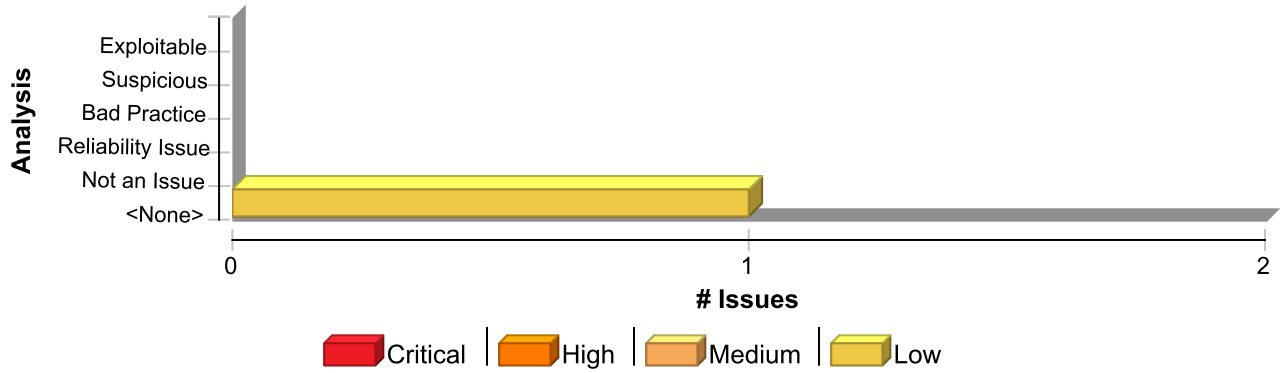
Null-pointer exceptions usually occur when one or more of the programmer's assumptions is violated. Specifically, dereference-after-check errors occur when a program makes an explicit check for `null`, but proceeds to dereference the object when it is known to be `null`. Errors of this type are often 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	1	0	0	1
Total	**1**	**0**	**0**	**1**

© Copyright [2008-2022] Micro Focus or one of its affiliates.

Redundant Null Check — Low

Package: akka.persistence.query

Offset.scala, line 49 (Redundant Null Check) — Low

Issue Details

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

Sink Details

Sink: Dereferenced : value
Enclosing Method: TimeBasedUUID()
File: Offset.scala:49
Taint Flags:

```
46 */
47 final case class TimeBasedUUID(value: UUID) extends Offset with Ordered[TimeBasedUUID] {
48 if (value == null || value.version != 1) {
49 throw new IllegalArgumentException("UUID " + value + " is not a time-based UUID")
50 }
51
52 override def compare(other: TimeBasedUUID): Int = UUIDComparator.comparator.compare(value, other.value)
```


© Copyright [2008-2022] Micro Focus or one of its affiliates.

© Copyright [2008-2022] Micro Focus or one of its affiliates.