Fortify Standalone Report Generator

Developer Workbook

akka-persistence

Table of Contents

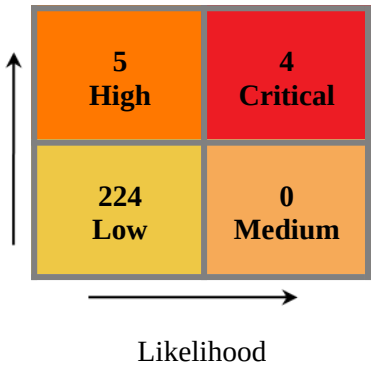
Executive Summary

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 project audit. The information contained in this workbook is targeted at project managers and developers.

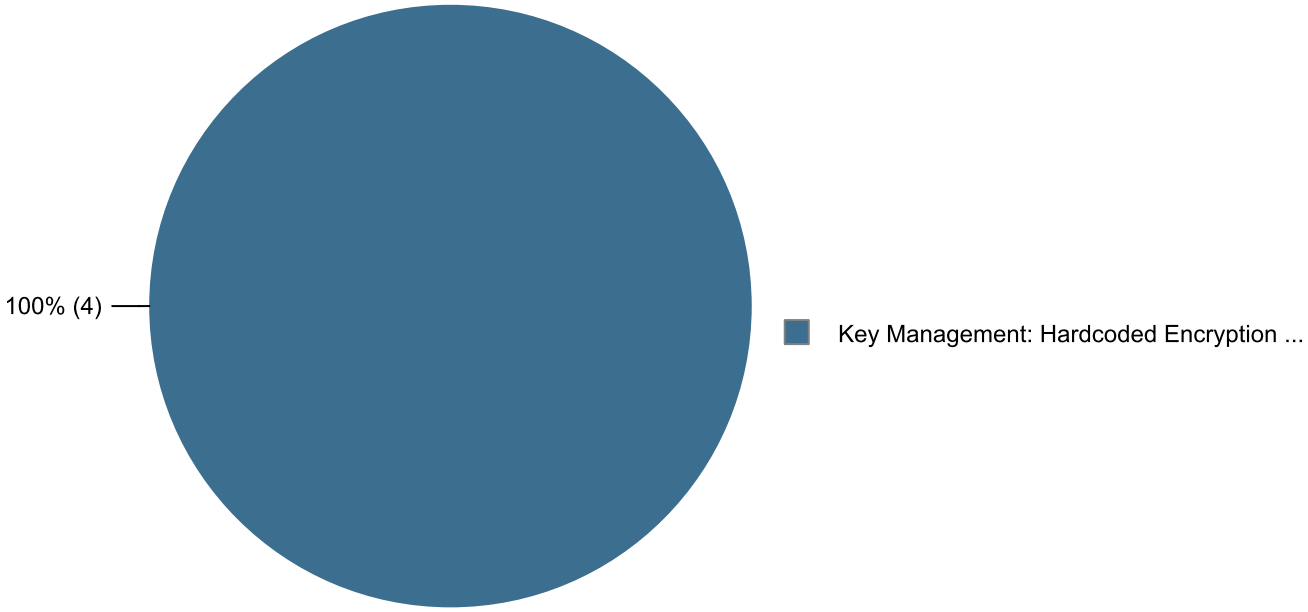
This section provides an overview of the issues uncovered during analysis.

Project Name:	akka-persistence
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:	Jun 16, 2022, 11:35 AM	**Engine Version:**	21.1.1.0009
Host Name:	Jacks-Work-MBP.local	**Certification:**	VALID
Number of Files:	84	**Lines of Code:**	7,540

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

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 / 47	0 / 47
Code Correctness: Erroneous String Compare	0	0	0	0 / 7	0 / 7
Code Correctness: Non-Static Inner Class Implements Serializable	0	0	0	0 / 112	0 / 112
Code Correctness: Non-Synchronized Method Overrides Synchronized Method	0	0	0	0 / 6	0 / 6
Dead Code: Expression is Always false	0	0	0	0 / 40	0 / 40
Dead Code: Expression is Always true	0	0	0	0 / 2	0 / 2
Denial of Service	0	0	0	0 / 1	0 / 1
Insecure Randomness	0	0 / 5	0	0	0 / 5
J2EE Bad Practices: Threads	0	0	0	0 / 3	0 / 3
Key Management: Hardcoded Encryption Key	0 / 4	0	0	0	0 / 4
Object Model Violation: Just one of equals() and hashCode() Defined	0	0	0	0 / 1	0 / 1
Poor Error Handling: Overly Broad Catch	0	0	0	0 / 1	0 / 1
System Information Leak	0	0	0	0 / 1	0 / 1
Unchecked Return Value	0	0	0	0 / 3	0 / 3

Results Outline

Code Correctness: Constructor Invokes Overridable Function (47 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");
    }
  }
}
```

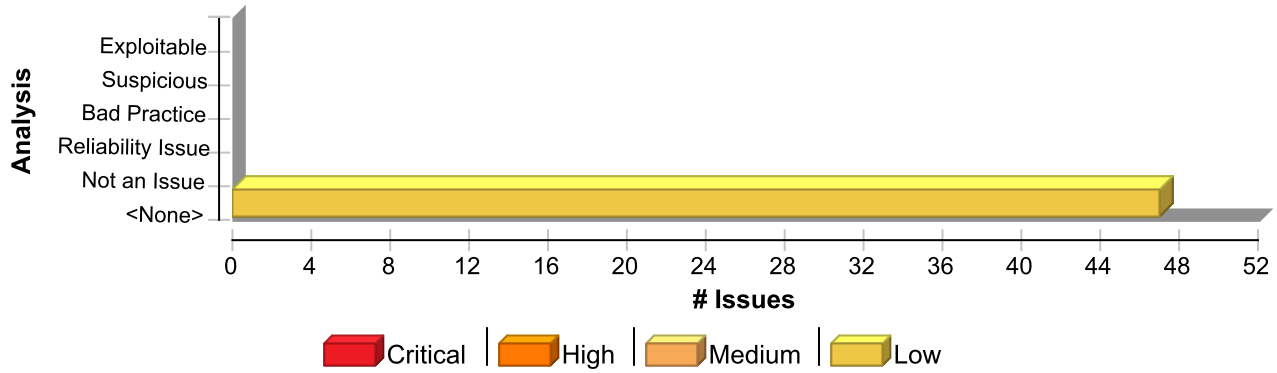
The code in `Example 1` prints "Attack successful!", since the `Attacker` class overrides the `validateUser()` function that is called from the constructor of the superclass `User`, and Java will first look in the subclass for functions called from the constructor.

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	47	0	0	47
Total	**47**	**0**	**0**	**47**

Code Correctness: Constructor Invokes Overridable Function	**Low**
Package: akka.persistence	
main/scala/akka/persistence/Persistence.scala, line 215 (Code Correctness: Constructor Invokes Overridable Function)	**Low**
Issue Details	

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

main/scala/akka/persistence/Persistence.scala, line 215 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: Persistence()
File: main/scala/akka/persistence/Persistence.scala:215
Taint Flags:

```
212 * of recoveries that can be in progress at the same time.
213 */
214 @InternalApi private[akka] val recoveryPermitter: ActorRef = {
215 val maxPermits = config.getInt("max-concurrent-recoveries")
216 system.systemActorOf(RecoveryPermitter.props(maxPermits), "recoveryPermitter")
217 }
218
```

main/scala/akka/persistence/Persistence.scala, line 252 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: Persistence()
File: main/scala/akka/persistence/Persistence.scala:252
Taint Flags:

```
249 .map(_.create(system.settings.config))
250 .get
251
252 val settings = new PersistenceSettings(config)
253
254 /** Discovered persistence journal and snapshot store plugins. */
255 private val pluginExtensionId = new AtomicReference[Map[String, ExtensionId[PluginHolder]]](Map.empty)
```

main/scala/akka/persistence/Persistence.scala, line 259 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

main/scala/akka/persistence/Persistence.scala, line 259 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: config
Enclosing Method: Persistence()
File: main/scala/akka/persistence/Persistence.scala:259
Taint Flags:

```
256
257 config
258 .getStringList("journal.auto-start-journals")
259 .forEach(new Consumer[String] {
260 override def accept(id: String): Unit = {
261 log.info(s"Auto-starting journal plugin `$id`")
262 journalFor(id)
```

main/scala/akka/persistence/Persistence.scala, line 267 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: Persistence()
File: main/scala/akka/persistence/Persistence.scala:267
Taint Flags:

```
264 })
265 config
266 .getStringList("snapshot-store.auto-start-snapshot-stores")
267 .forEach(new Consumer[String] {
268 override def accept(id: String): Unit = {
269 log.info(s"Auto-starting snapshot store `$id`")
270 snapshotStoreFor(id)
```

test/scala/akka/persistence/EndToEndEventAdapterSpec.scala, line 132 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: journalName
Enclosing Method: EndToEndEventAdapterSpec()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

test/scala/akka/persistence/EndToEndEventAdapterSpec.scala, line 132 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: test/scala/akka/persistence/EndToEndEventAdapterSpec.scala:132
Taint Flags:

```
129
130 val noAdaptersConfig = ConfigFactory.parseString("")
131
132 val adaptersConfig = ConfigFactory.parseString(s"""
133 |akka.persistence.journal {
134 | $journalName {
135 | event-adapters {
```

test/scala/akka/persistence/EndToEndEventAdapterSpec.scala, line 152 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: journalName
Enclosing Method: EndToEndEventAdapterSpec()
File: test/scala/akka/persistence/EndToEndEventAdapterSpec.scala:152
Taint Flags:

```
149 |akka.loggers = ["akka.testkit.TestEventListener"]
150 """.stripMargin)
151
152 val newAdaptersConfig = ConfigFactory.parseString(s"""
153 |akka.persistence.journal {
154 | $journalName {
155 | event-adapters {
```

test/scala/akka/persistence/PersistentActorRecoveryTimeoutSpec.scala, line 71 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: PersistentActorRecoveryTimeoutSpec()
File: test/scala/akka/persistence/PersistentActorRecoveryTimeoutSpec.scala:71
Taint Flags:

```
68 }
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorRecoveryTimeoutSpec.scala, line 71 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
69
70 class PersistentActorRecoveryTimeoutSpec
71  extends AkkaSpec(PersistentActorRecoveryTimeoutSpec.config)
72  with ImplicitSender {
73
74  import PersistentActorRecoveryTimeoutSpec.journalId
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 91 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: probe
Enclosing Method: JournalProbe()
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:91
Taint Flags:

```
88 }
89 class JournalProbe(implicit private val system: ExtendedActorSystem) extends Extension {
90  val probe = TestProbe()
91  val ref = probe.ref
92 }
93
94 class JournalPuppet extends Actor {
```

test/scala/akka/persistence/PersistentActorBoundedStashingSpec.scala, line 47 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: capacity
Enclosing Method: PersistentActorBoundedStashingSpec()
File: test/scala/akka/persistence/PersistentActorBoundedStashingSpec.scala:47
Taint Flags:

```
44
45  val capacity = 10
46
47  val templateConfig =
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorBoundedStashingSpec.scala, line 47 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
48 s"""
49 |akka.actor.default-mailbox.stash-capacity=$capacity
50 |akka.actor.guardian-supervisor-strategy="akka.actor.StoppingSupervisorStrategy"
```

test/scala/akka/persistence/PersistentActorBoundedStashingSpec.scala, line 54 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: templateConfig
Enclosing Method: PersistentActorBoundedStashingSpec()
File: test/scala/akka/persistence/PersistentActorBoundedStashingSpec.scala:54
Taint Flags:

```
51 |akka.persistence.internal-stash-overflow-strategy = "%s"
52 |""".stripMargin
53
54 val throwConfig = String.format(templateConfig, "akka.persistence.ThrowExceptionConfigurator")
55 val discardConfig = String.format(templateConfig, "akka.persistence.DiscardConfigurator")
56 val replyToConfig =
57 String.format(templateConfig, "akka.persistence.PersistentActorBoundedStashingSpec$ReplyToWithRejectConfigurator")
```

test/scala/akka/persistence/PersistentActorBoundedStashingSpec.scala, line 55 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: templateConfig
Enclosing Method: PersistentActorBoundedStashingSpec()
File: test/scala/akka/persistence/PersistentActorBoundedStashingSpec.scala:55
Taint Flags:

```
52 |""".stripMargin
53
54 val throwConfig = String.format(templateConfig, "akka.persistence.ThrowExceptionConfigurator")
55 val discardConfig = String.format(templateConfig, "akka.persistence.DiscardConfigurator")
56 val replyToConfig =
57 String.format(templateConfig, "akka.persistence.PersistentActorBoundedStashingSpec$ReplyToWithRejectConfigurator")
58
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorBoundedStashingSpec.scala, line 56 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: templateConfig
Enclosing Method: PersistentActorBoundedStashingSpec()
File: test/scala/akka/persistence/PersistentActorBoundedStashingSpec.scala:56
Taint Flags:

```
53
54  val throwConfig = String.format(templateConfig, "akka.persistence.ThrowExceptionConfigurator")
55  val discardConfig = String.format(templateConfig, "akka.persistence.DiscardConfigurator")
56  val replyToConfig =
57  String.format(templateConfig, "akka.persistence.PersistentActorBoundedStashingSpec$ReplyToWithRejectConfigurator")
58
59 }
```

main/scala/akka/persistence/PersistentActor.scala, line 102 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply$default$3
Enclosing Method: Recovery()
File: main/scala/akka/persistence/PersistentActor.scala:102
Taint Flags:

```
99  *
100  * @see [[Recovery]]
101  */
102  val none: Recovery = Recovery(toSequenceNr = 0L, fromSnapshot = SnapshotSelectionCriteria.None)
103
104 }
105
```

main/scala/akka/persistence/Eventsourced.scala, line 81 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

main/scala/akka/persistence/Eventsourced.scala, line 81 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: akka$persistence$Eventsourced$$instanceIdCounter
Enclosing Method: Eventsourced()
File: main/scala/akka/persistence/Eventsourced.scala:81
Taint Flags:

```
78 extension.snapshotStoreFor(snapshotPluginId, snapshotPluginConfig)
79 }
80
81 private val instanceId: Int = Eventsourced.instanceIdCounter.getAndIncrement()
82 private val writerUuid = UUID.randomUUID.toString
83
84 private var journalBatch = Vector.empty[PersistentEnvelope]
```

main/scala/akka/persistence/SnapshotProtocol.scala, line 190 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply$default$4
Enclosing Method: SnapshotSelectionCriteria()
File: main/scala/akka/persistence/SnapshotProtocol.scala:190
Taint Flags:

```
187 /**
188 * The latest saved snapshot.
189 */
190 val Latest = SnapshotSelectionCriteria()
191
192 /**
193 * No saved snapshot matches.
```

main/scala/akka/persistence/SnapshotProtocol.scala, line 195 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply$default$4
Enclosing Method: SnapshotSelectionCriteria()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

main/scala/akka/persistence/SnapshotProtocol.scala, line 195 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: main/scala/akka/persistence/SnapshotProtocol.scala:195
Taint Flags:

```
192 /**
193 * No saved snapshot matches.
194 */
195 val None = SnapshotSelectionCriteria(0L, 0L)
196
197 /**
198 * Java API.
```

main/scala/akka/persistence/SnapshotProtocol.scala, line 190 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply$default$3
Enclosing Method: SnapshotSelectionCriteria()
File: main/scala/akka/persistence/SnapshotProtocol.scala:190
Taint Flags:

```
187 /**
188 * The latest saved snapshot.
189 */
190 val Latest = SnapshotSelectionCriteria()
191
192 /**
193 * No saved snapshot matches.
```

main/scala/akka/persistence/SnapshotProtocol.scala, line 195 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply$default$3
Enclosing Method: SnapshotSelectionCriteria()
File: main/scala/akka/persistence/SnapshotProtocol.scala:195
Taint Flags:

```
192 /**
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

main/scala/akka/persistence/SnapshotProtocol.scala, line 195 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
193  * No saved snapshot matches.
194  */
195  val None = SnapshotSelectionCriteria(0L, 0L)
196
197  /**
198  * Java API.
```

test/scala/akka/persistence/SnapshotRecoveryLocalStoreSpec.scala, line 12 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: persistenceId
Enclosing Method: SnapshotRecoveryLocalStoreSpec()
File: test/scala/akka/persistence/SnapshotRecoveryLocalStoreSpec.scala:12
Taint Flags:

```
9
10 object SnapshotRecoveryLocalStoreSpec {
11  val persistenceId = "europe"
12  val extendedName = persistenceId + "italy"
13
14  case object TakeSnapshot
15
```

main/scala/akka/persistence/SnapshotProtocol.scala, line 190 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply$default$2
Enclosing Method: SnapshotSelectionCriteria()
File: main/scala/akka/persistence/SnapshotProtocol.scala:190
Taint Flags:

```
187  /**
188  * The latest saved snapshot.
189  */
190  val Latest = SnapshotSelectionCriteria()
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.persistence

main/scala/akka/persistence/SnapshotProtocol.scala, line 190 (Code Correctness: Constructor Invokes Overridable Function) **Low**

```
191
192 /**
193 * No saved snapshot matches.
```

main/scala/akka/persistence/SnapshotProtocol.scala, line 190 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply$default$1
Enclosing Method: SnapshotSelectionCriteria()
File: main/scala/akka/persistence/SnapshotProtocol.scala:190
Taint Flags:

```
187 /**
188 * The latest saved snapshot.
189 */
190 val Latest = SnapshotSelectionCriteria()
191
192 /**
193 * No saved snapshot matches.
```

test/scala/akka/persistence/EventAdapterSpec.scala, line 107 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: DomainEventClassName
Enclosing Method: EventAdapterSpec()
File: test/scala/akka/persistence/EventAdapterSpec.scala:107
Taint Flags:

```
104 import EventAdapterSpec._
105
106 def this() =
107 this(
108 "inmem",
109 PersistenceSpec.config("inmem", "InmemPersistentTaggingSpec"),
110 ConfigFactory.parseString(s"""
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

test/scala/akka/persistence/EventAdapterSpec.scala, line 107 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: JournalModelClassName
Enclosing Method: EventAdapterSpec()
File: test/scala/akka/persistence/EventAdapterSpec.scala:107
Taint Flags:

```
104 import EventAdapterSpec._
105
106 def this() =
107 this(
108 "inmem",
109 PersistenceSpec.config("inmem", "InmemPersistentTaggingSpec"),
110 ConfigFactory.parseString(s"""
```

test/scala/akka/persistence/SnapshotDirectoryFailureSpec.scala, line 36 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: inUseSnapshotPath
Enclosing Method: SnapshotDirectoryFailureSpec()
File: test/scala/akka/persistence/SnapshotDirectoryFailureSpec.scala:36
Taint Flags:

```
33 PersistenceSpec.config(
34 "inmem",
35 "SnapshotDirectoryFailureSpec",
36 extraConfig = Some(s"""
37 akka.persistence.snapshot-store.local.dir = "${SnapshotDirectoryFailureSpec.inUseSnapshotPath}"
38 """)))
39 with ImplicitSender {
```

test/scala/akka/persistence/SnapshotDirectoryFailureSpec.scala, line 43 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotDirectoryFailureSpec.scala, line 43 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: inUseSnapshotPath
Enclosing Method: SnapshotDirectoryFailureSpec()
File: test/scala/akka/persistence/SnapshotDirectoryFailureSpec.scala:43
Taint Flags:

```
40
41 import SnapshotDirectoryFailureSpec._
42
43 val file = new File(inUseSnapshotPath)
44
45 override protected def atStartup(): Unit = {
46 if (!file.createNewFile()) throw new IOException(s"Failed to create test file [${file.getCanonicalFile}]")
```

test/scala/akka/persistence/SnapshotRecoveryWithEmptyJournalSpec.scala, line 68 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: survivingSnapshotPath
Enclosing Method: SnapshotRecoveryWithEmptyJournalSpec()
File: test/scala/akka/persistence/SnapshotRecoveryWithEmptyJournalSpec.scala:68
Taint Flags:

```
65 PersistenceSpec.config(
66 "inmem",
67 "SnapshotRecoveryWithEmptyJournalSpec",
68 extraConfig = Some(s"""
69 akka.persistence.snapshot-store.local.dir = "${SnapshotRecoveryWithEmptyJournalSpec.survivingSnapshotPath}"
70 """))
71 with ImplicitSender {
```

test/scala/akka/persistence/SnapshotRecoveryWithEmptyJournalSpec.scala, line 76 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: survivingSnapshotPath
Enclosing Method: SnapshotRecoveryWithEmptyJournalSpec()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotRecoveryWithEmptyJournalSpec.scala, line 76 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: test/scala/akka/persistence/SnapshotRecoveryWithEmptyJournalSpec.scala:76
Taint Flags:

```
73 import SnapshotRecoveryWithEmptyJournalSpec._
74
75 val persistenceId: String = namePrefix
76 val snapshotsDir: File = new File(survivingSnapshotPath)
77
78 val serializationExtension: Serialization = SerializationExtension(system)
79
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 103 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: PersistentActorJournalProtocolSpec()
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:103
Taint Flags:

```
100
101 import PersistentActorJournalProtocolSpec._
102
103 class PersistentActorJournalProtocolSpec extends AkkaSpec(config) with ImplicitSender {
104
105 val journal = JournalPuppet(system).probe
106
```

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 172 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: AtLeastOnceDeliveryFailureSpec()
File: test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala:172
Taint Flags:

```
169 }
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 172 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
170
171 class AtLeastOnceDeliveryFailureSpec
172  extends AkkaSpec(AtLeastOnceDeliveryFailureSpec.config)
173  with Cleanup
174  with ImplicitSender {
175  import AtLeastOnceDeliveryFailureSpec._
```

test/scala/akka/persistence/PerformanceSpec.scala, line 117 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: PerformanceSpec()
File: test/scala/akka/persistence/PerformanceSpec.scala:117
Taint Flags:

```
114 }
115
116 class PerformanceSpec
117  extends PersistenceSpec(
118  PersistenceSpec
119  .config("inmem", "PerformanceSpec", serialization = "off")
120  .withFallback(ConfigFactory.parseString(PerformanceSpec.config)))
```

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 42 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: key
Enclosing Method: SnapshotAfter()
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:42
Taint Flags:

```
39  */
40 private[akka] class SnapshotAfter(config: Config) extends Extension {
41  val key = "akka.persistence.fsm.snapshot-after"
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 42 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
42 val snapshotAfterValue = config.getString(key).toLowerCase match {
43 case "off" => None
44 case _ => Some(config.getInt(key))
45 }
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 44 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: key
Enclosing Method: SnapshotAfter()
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:44
Taint Flags:

```
41 val key = "akka.persistence.fsm.snapshot-after"
42 val snapshotAfterValue = config.getString(key).toLowerCase match {
43 case "off" => None
44 case _ => Some(config.getInt(key))
45 }
46
47 /**
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 51 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: snapshotAfterValue
Enclosing Method: SnapshotAfter()
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:51
Taint Flags:

```
48 * Function that takes lastSequenceNr as the param, and returns whether the passed
49 * sequence number should trigger auto snapshot or not
50 */
51 val isSnapshotAfterSeqNo: Long => Boolean = snapshotAfterValue match {
52 case Some(snapShotAfterValue) => (seqNo: Long) => seqNo % snapShotAfterValue == 0
53 case None => (_: Long) => false //always false, if snapshotAfter is not specified in config
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 51 (Code Correctness: Constructor Invokes Overridable Function)	Low

```
54 }
```

Package: akka.persistence.journal

test/scala/akka/persistence/journal/ReplayFilterSpec.scala, line 22 (Code Correctness: Constructor Invokes Overridable Function)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: writerA
Enclosing Method: ReplayFilterSpec()
File: test/scala/akka/persistence/journal/ReplayFilterSpec.scala:22
Taint Flags:

```
19 val n1 = ReplayedMessage(PersistentRepr("a", 13, "p1", "", writerUuid = PersistentRepr.Undefined))
20 val n2 = ReplayedMessage(PersistentRepr("b", 14, "p1", "", writerUuid = PersistentRepr.Undefined))
21
22 val m1 = ReplayedMessage(PersistentRepr("a", 13, "p1", "", writerUuid = writerA))
23 val m2 = ReplayedMessage(PersistentRepr("b", 14, "p1", "", writerUuid = writerA))
24 val m3 = ReplayedMessage(PersistentRepr("c", 15, "p1", "", writerUuid = writerA))
25 val m4 = ReplayedMessage(PersistentRepr("d", 16, "p1", "", writerUuid = writerA))
```

test/scala/akka/persistence/journal/ReplayFilterSpec.scala, line 23 (Code Correctness: Constructor Invokes Overridable Function)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: writerA
Enclosing Method: ReplayFilterSpec()
File: test/scala/akka/persistence/journal/ReplayFilterSpec.scala:23
Taint Flags:

```
20 val n2 = ReplayedMessage(PersistentRepr("b", 14, "p1", "", writerUuid = PersistentRepr.Undefined))
21
22 val m1 = ReplayedMessage(PersistentRepr("a", 13, "p1", "", writerUuid = writerA))
23 val m2 = ReplayedMessage(PersistentRepr("b", 14, "p1", "", writerUuid = writerA))
24 val m3 = ReplayedMessage(PersistentRepr("c", 15, "p1", "", writerUuid = writerA))
25 val m4 = ReplayedMessage(PersistentRepr("d", 16, "p1", "", writerUuid = writerA))
26 val successMsg = RecoverySuccess(15)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence.journal

test/scala/akka/persistence/journal/ReplayFilterSpec.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: writerA
Enclosing Method: ReplayFilterSpec()
File: test/scala/akka/persistence/journal/ReplayFilterSpec.scala:24
Taint Flags:

```
21
22 val m1 = ReplayedMessage(PersistentRepr("a", 13, "p1", "", writerUuid = writerA))
23 val m2 = ReplayedMessage(PersistentRepr("b", 14, "p1", "", writerUuid = writerA))
24 val m3 = ReplayedMessage(PersistentRepr("c", 15, "p1", "", writerUuid = writerA))
25 val m4 = ReplayedMessage(PersistentRepr("d", 16, "p1", "", writerUuid = writerA))
26 val successMsg = RecoverySuccess(15)
27
```

test/scala/akka/persistence/journal/ReplayFilterSpec.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: writerA
Enclosing Method: ReplayFilterSpec()
File: test/scala/akka/persistence/journal/ReplayFilterSpec.scala:25
Taint Flags:

```
22 val m1 = ReplayedMessage(PersistentRepr("a", 13, "p1", "", writerUuid = writerA))
23 val m2 = ReplayedMessage(PersistentRepr("b", 14, "p1", "", writerUuid = writerA))
24 val m3 = ReplayedMessage(PersistentRepr("c", 15, "p1", "", writerUuid = writerA))
25 val m4 = ReplayedMessage(PersistentRepr("d", 16, "p1", "", writerUuid = writerA))
26 val successMsg = RecoverySuccess(15)
27
28 "ReplayFilter in RepairByDiscardOld mode" must {
```

test/scala/akka/persistence/journal/InmemEventAdaptersSpec.scala, line 49 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence.journal

test/scala/akka/persistence/journal/InmemEventAdaptersSpec.scala, line 49 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: config
Enclosing Method: InmemEventAdaptersSpec()
File: test/scala/akka/persistence/journal/InmemEventAdaptersSpec.scala:49
Taint Flags:

```
46 """.stripMargin).withFallback(ConfigFactory.load())
47
48 val extendedActorSystem = system.asInstanceOf[ExtendedActorSystem]
49 val inmemConfig = config.getConfig("akka.persistence.journal.inmem")
50
51 "EventAdapters" must {
52 "parse configuration and resolve adapter definitions" in {
```

main/scala/akka/persistence/journal/PersistencePluginProxy.scala, line 90 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akka$persistence$journal$PersistencePluginProxy$$pluginType
Enclosing Method: PersistencePluginProxy()
File: main/scala/akka/persistence/journal/PersistencePluginProxy.scala:90
Taint Flags:

```
87
88 private val initTimeout: FiniteDuration = config.getDuration("init-timeout", MILLISECONDS).millis
89 private val targetPluginId: String = {
90 val key = s"target-${pluginType.qualifier}-plugin"
91 config.getString(key).requiring(_ != "", s"$pluginId.$key must be defined")
92 }
93 private val startTarget: Boolean = config.getBoolean(s"start-target-${pluginType.qualifier}")
```

main/scala/akka/persistence/journal/PersistencePluginProxy.scala, line 93 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akka$persistence$journal$PersistencePluginProxy$$pluginType
Enclosing Method: PersistencePluginProxy()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence.journal

main/scala/akka/persistence/journal/PersistencePluginProxy.scala, line 93 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: main/scala/akka/persistence/journal/PersistencePluginProxy.scala:93
Taint Flags:

```
90 val key = s"target-${pluginType.qualifier}-plugin"
91 config.getString(key).requiring(_ != "", s"$pluginId.$key must be defined")
92 }
93 private val startTarget: Boolean = config.getBoolean(s"start-target-${pluginType.qualifier}")
94
95 override def preStart(): Unit = {
96 if (startTarget) {
```

main/scala/akka/persistence/journal/PersistencePluginProxy.scala, line 81 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: pluginId
Enclosing Method: PersistencePluginProxy()
File: main/scala/akka/persistence/journal/PersistencePluginProxy.scala:81
Taint Flags:

```
78 import SnapshotProtocol._
79
80 private val pluginId = self.path.name
81 private val pluginType: PluginType = pluginId match {
82 case "akka.persistence.journal.proxy" => Journal
83 case "akka.persistence.snapshot-store.proxy" => SnapshotStore
84 case other =>
```

Package: akka.persistence.journal.chaos

test/scala/akka/persistence/journal/chaos/ChaosJournal.scala, line 36 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: ChaosJournal()
File: test/scala/akka/persistence/journal/chaos/ChaosJournal.scala:36
Taint Flags:

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence.journal.chaos

test/scala/akka/persistence/journal/chaos/ChaosJournal.scala, line 36 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
33 import ChaosJournalMessages.{ delete => del, _ }
34
35 val config = context.system.settings.config.getConfig("akka.persistence.journal.chaos")
36 val writeFailureRate = config.getDouble("write-failure-rate")
37 val deleteFailureRate = config.getDouble("delete-failure-rate")
38 val replayFailureRate = config.getDouble("replay-failure-rate")
39 val readHighestFailureRate = config.getDouble("read-highest-failure-rate")
```

test/scala/akka/persistence/journal/chaos/ChaosJournal.scala, line 37 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: ChaosJournal()
File: test/scala/akka/persistence/journal/chaos/ChaosJournal.scala:37
Taint Flags:

```
34
35 val config = context.system.settings.config.getConfig("akka.persistence.journal.chaos")
36 val writeFailureRate = config.getDouble("write-failure-rate")
37 val deleteFailureRate = config.getDouble("delete-failure-rate")
38 val replayFailureRate = config.getDouble("replay-failure-rate")
39 val readHighestFailureRate = config.getDouble("read-highest-failure-rate")
40
```

test/scala/akka/persistence/journal/chaos/ChaosJournal.scala, line 38 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: ChaosJournal()
File: test/scala/akka/persistence/journal/chaos/ChaosJournal.scala:38
Taint Flags:

```
35 val config = context.system.settings.config.getConfig("akka.persistence.journal.chaos")
36 val writeFailureRate = config.getDouble("write-failure-rate")
37 val deleteFailureRate = config.getDouble("delete-failure-rate")
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.persistence.journal.chaos

test/scala/akka/persistence/journal/chaos/ChaosJournal.scala, line 38 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
38 val replayFailureRate = config.getDouble("replay-failure-rate")
39 val readHighestFailureRate = config.getDouble("read-highest-failure-rate")
40
41 def random = ThreadLocalRandom.current
```

test/scala/akka/persistence/journal/chaos/ChaosJournal.scala, line 39 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: ChaosJournal()
File: test/scala/akka/persistence/journal/chaos/ChaosJournal.scala:39
Taint Flags:

```
36 val writeFailureRate = config.getDouble("write-failure-rate")
37 val deleteFailureRate = config.getDouble("delete-failure-rate")
38 val replayFailureRate = config.getDouble("replay-failure-rate")
39 val readHighestFailureRate = config.getDouble("read-highest-failure-rate")
40
41 def random = ThreadLocalRandom.current
42
```

Package: akka.persistence.journal.leveldb

main/scala/akka/persistence/journal/leveldb/LeveldbStore.scala, line 53 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: toCompactionIntervalMap
Enclosing Method: LeveldbStore()
File: main/scala/akka/persistence/journal/leveldb/LeveldbStore.scala:53
Taint Flags:

```
50 val leveldbWriteOptions = new WriteOptions().sync(config.getBoolean("fsync")).snapshot(false)
51 val leveldbDir = new File(config.getString("dir"))
52 var leveldb: DB = _
53 override val compactionIntervals: Map[String, Long] =
54 LeveldbStore.toCompactionIntervalMap(config.getObject("compaction-intervals"))
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.persistence.journal.leveldb

main/scala/akka/persistence/journal/leveldb/LeveldbStore.scala, line 53 (Code Correctness: Constructor Invokes Overridable Function)	Low

```
55
56 import scala.annotation.nowarn
```

Package: akka.persistence.state

main/scala/akka/persistence/state/DurableStateStoreRegistry.scala, line 47 (Code Correctness: Constructor Invokes Overridable Function)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: pluginProvider
Enclosing Method: DurableStateStoreRegistry()
File: main/scala/akka/persistence/state/DurableStateStoreRegistry.scala:47
Taint Flags:

```
44 }
45
46 class DurableStateStoreRegistry(system: ExtendedActorSystem)
47  extends PersistencePlugin[scaladsl.DurableStateStore[_], javadsl.DurableStateStore[_], DurableStateStoreProvider](
48  system)(ClassTag(classOf[DurableStateStoreProvider]), DurableStateStoreRegistry.pluginProvider)
49  with Extension {
50
```

Code Correctness: Erroneous String Compare (7 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");
  }
```

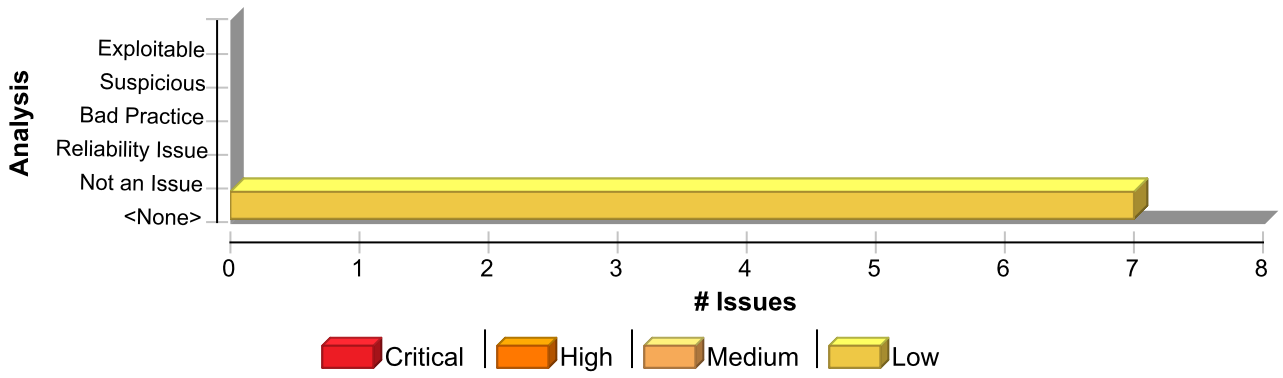
The `==` and `!=` operators will only behave as expected when they are used to compare strings contained in objects that are equal. The most common way for this to occur is for the strings to be interned, whereby the strings are added to a pool of objects maintained by the `String` class. Once a string 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	7	0	0	7
Total	**7**	**0**	**0**	**7**

Code Correctness: Erroneous String Compare — Low

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 42 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: SnapshotAfter()
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:42
Taint Flags:

```
39 */
40 private[akka] class SnapshotAfter(config: Config) extends Extension {
41  val key = "akka.persistence.fsm.snapshot-after"
42  val snapshotAfterValue = config.getString(key).toLowerCase match {
43  case "off" => None
44  case _ => Some(config.getInt(key))
45  }
```

Package: akka.persistence.journal

main/scala/akka/persistence/journal/PersistencePluginProxy.scala, line 81 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: PersistencePluginProxy()
File: main/scala/akka/persistence/journal/PersistencePluginProxy.scala:81
Taint Flags:

```
78 import SnapshotProtocol._
79
80 private val pluginId = self.path.name
81 private val pluginType: PluginType = pluginId match {
82  case "akka.persistence.journal.proxy" => Journal
83  case "akka.persistence.snapshot-store.proxy" => SnapshotStore
84  case other =>
```

main/scala/akka/persistence/journal/AsyncWriteJournal.scala, line 39 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality

Code Correctness: Erroneous String Compare — Low

Package: akka.persistence.journal

main/scala/akka/persistence/journal/AsyncWriteJournal.scala, line 39 (Code Correctness: Erroneous String Compare) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: AsyncWriteJournal()
File: main/scala/akka/persistence/journal/AsyncWriteJournal.scala:39
Taint Flags:

```
36 }
37
38 private val replayFilterMode: ReplayFilter.Mode =
39 toRootLowerCase(config.getString("replay-filter.mode")) match {
40 case "off" => ReplayFilter.Disabled
41 case "repair-by-discard-old" => ReplayFilter.RepairByDiscardOld
42 case "fail" => ReplayFilter.Fail
```

main/scala/akka/persistence/journal/AsyncWriteJournal.scala, line 39 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: AsyncWriteJournal()
File: main/scala/akka/persistence/journal/AsyncWriteJournal.scala:39
Taint Flags:

```
36 }
37
38 private val replayFilterMode: ReplayFilter.Mode =
39 toRootLowerCase(config.getString("replay-filter.mode")) match {
40 case "off" => ReplayFilter.Disabled
41 case "repair-by-discard-old" => ReplayFilter.RepairByDiscardOld
42 case "fail" => ReplayFilter.Fail
```

main/scala/akka/persistence/journal/AsyncWriteJournal.scala, line 39 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Erroneous String Compare — Low

Package: akka.persistence.journal

main/scala/akka/persistence/journal/AsyncWriteJournal.scala, line 39 (Code Correctness: Erroneous String Compare) — Low

Sink: Operation
Enclosing Method: AsyncWriteJournal()
File: main/scala/akka/persistence/journal/AsyncWriteJournal.scala:39
Taint Flags:

```
36 }
37
38 private val replayFilterMode: ReplayFilter.Mode =
39 toRootLowerCase(config.getString("replay-filter.mode")) match {
40 case "off" => ReplayFilter.Disabled
41 case "repair-by-discard-old" => ReplayFilter.RepairByDiscardOld
42 case "fail" => ReplayFilter.Fail
```

main/scala/akka/persistence/journal/PersistencePluginProxy.scala, line 81 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: PersistencePluginProxy()
File: main/scala/akka/persistence/journal/PersistencePluginProxy.scala:81
Taint Flags:

```
78 import SnapshotProtocol._
79
80 private val pluginId = self.path.name
81 private val pluginType: PluginType = pluginId match {
82 case "akka.persistence.journal.proxy" => Journal
83 case "akka.persistence.snapshot-store.proxy" => SnapshotStore
84 case other =>
```

main/scala/akka/persistence/journal/AsyncWriteJournal.scala, line 39 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: AsyncWriteJournal()
File: main/scala/akka/persistence/journal/AsyncWriteJournal.scala:39
Taint Flags:

Code Correctness: Erroneous String Compare	Low

Package: akka.persistence.journal

main/scala/akka/persistence/journal/AsyncWriteJournal.scala, line 39 (Code Correctness: Erroneous String Compare)	Low

```
36  }
37
38  private val replayFilterMode: ReplayFilter.Mode =
39  toRootLowerCase(config.getString("replay-filter.mode")) match {
40  case "off" => ReplayFilter.Disabled
41  case "repair-by-discard-old" => ReplayFilter.RepairByDiscardOld
42  case "fail" => ReplayFilter.Fail
```

Code Correctness: Non-Static Inner Class Implements Serializable (112 issues)

Abstract

Inner classes implementing `java.io.Serializable` may cause problems and leak information from the outer class.

Explanation

Serialization of inner classes lead to serialization of the outer class, therefore possibly leaking information or leading to a runtime error if the outer class is not serializable. As well as this, serializing inner classes may cause platform dependencies since the Java compiler creates synthetic fields in order to implement inner classes, but these are implementation dependent, and may vary from compiler to compiler. **Example 1:** The following code allows serialization of an inner class.

```
...
class User implements Serializable {
  private int accessLevel;
  class Registrator implements Serializable {
    ...
  }
}
```

In `Example 1`, when the inner class `Registrator` is serialized, it will also serialize the field `accessLevel` from the outer class `User`.

Recommendation

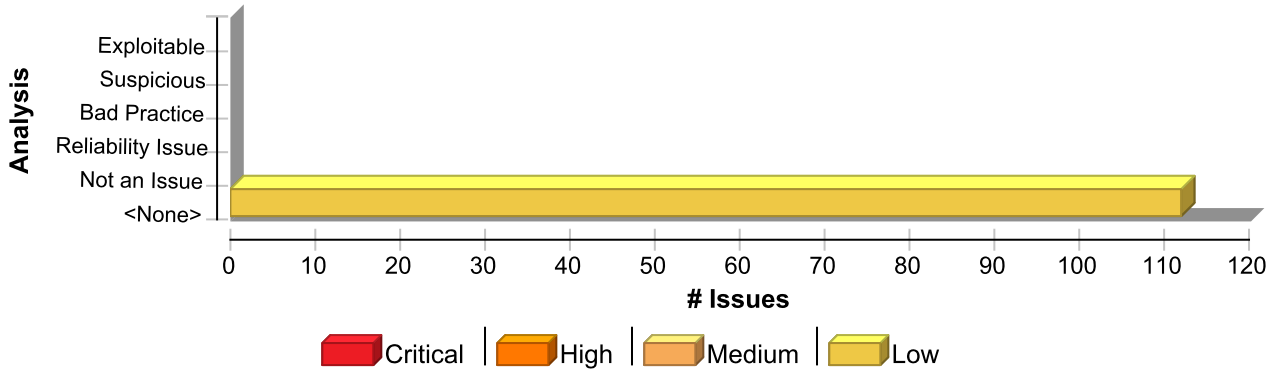
When using inner classes, they should not be serialized, or they should be changed to static-nested classes, since these do not have the drawbacks that non-static inner classes have when serialized. When a nested class is static it inherently has no association with instance variables (including those of the outer class), and would not cause serialization of the outer class. **Example 2:** The following code changes the example in `Example 1`, by stopping the inner class from implementing `java.io.Serializable`.

```
...
class User implements Serializable {
  private int accessLevel;
  class Registrator {
    ...
  }
}
```

Example 2: The following code changes the example in `Example 1`, by making the inner class into a static-nested class.

```
...
class User implements Serializable {
  private int accessLevel;
  static class Registrator implements Serializable {
    ...
  }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Non-Static Inner Class Implements Serializable	112	0	0	112
Total	**112**	**0**	**0**	**112**

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 40 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliveryFailureSpec$Msg
File: test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala:40
Taint Flags:

```
37
38 case class Ack(i: Int)
39
40 case class Msg(deliveryId: Long, i: Int)
41 case class Confirm(deliveryId: Long, i: Int)
42
43 sealed trait Evt
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 20 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorStashingSpec$Evt

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 20 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:20
Taint Flags:

```
17
18 object PersistentActorStashingSpec {
19  final case class Cmd(data: Any)
20  final case class Evt(data: Any)
21
22  abstract class StashExamplePersistentActor(name: String) extends NamedPersistentActor(name) {
23  var events: List[Any] = Nil
```

test/scala/akka/persistence/EventAdapterSpec.scala, line 30 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventAdapterSpec$TaggedDataChanged
File: test/scala/akka/persistence/EventAdapterSpec.scala:30
Taint Flags:

```
27
28 final val DomainEventClassName = classOf[EventAdapterSpec].getCanonicalName + "$" + classOf[DomainEvent].getSimpleName
29 trait DomainEvent
30 final case class TaggedDataChanged(tags: immutable.Set[String], value: Int) extends DomainEvent
31 final case class UserDataChanged(countryCode: String, age: Int) extends DomainEvent
32
33 class UserAgeTaggingAdapter extends EventAdapter {
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 26 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorSpec$Evt
File: test/scala/akka/persistence/PersistentActorSpec.scala:26
Taint Flags:

```
23
24 final case class Cmd(data: Any)
25
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorSpec.scala, line 26 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
26 final case class Evt(data: Any)
27
28 final case class LatchCmd(latch: TestLatch, data: Any) extends NoSerializationVerificationNeeded
29
```

test/scala/akka/persistence/PerformanceSpec.scala, line 23 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PerformanceSpec$FailAt
File: test/scala/akka/persistence/PerformanceSpec.scala:23
Taint Flags:

```
20 """
21
22 case object StopMeasure
23 final case class FailAt(sequenceNr: Long)
24
25 class Measure(numberOfMessages: Int) {
26 private val NanoToSecond = 1000.0 * 1000 * 1000
```

main/scala/akka/persistence/Persistence.scala, line 157 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Persistence$PluginHolder
File: main/scala/akka/persistence/Persistence.scala:157
Taint Flags:

```
154 def lookup = Persistence
155
156 /** INTERNAL API. */
157 private[persistence] case class PluginHolder(actorFactory: () => ActorRef, adapters: EventAdapters, config: Config)
158 extends Extension {
159 // lazy creation of actor so that it's not started when only looking up adapters
160 lazy val actor: ActorRef = actorFactory()
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/EndToEndEventAdapterSpec.scala, line 26 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EndToEndEventAdapterSpec$A
File: test/scala/akka/persistence/EndToEndEventAdapterSpec.scala:26
Taint Flags:

```
23 object EndToEndEventAdapterSpec {
24
25  trait AppModel { def payload: Any }
26  case class A(payload: Any) extends AppModel
27  case class B(payload: Any) extends AppModel
28  case class NewA(payload: Any) extends AppModel
29  case class NewB(payload: Any) extends AppModel
```

test/scala/akka/persistence/OptimizedRecoverySpec.scala, line 16 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: OptimizedRecoverySpec$TestPersistentActor$Saved
File: test/scala/akka/persistence/OptimizedRecoverySpec.scala:16
Taint Flags:

```
13  object TestPersistentActor {
14  case object TakeSnapshot
15  final case class Save(s: String)
16  final case class Saved(s: String, seqNr: Long)
17  case object PersistFromRecoveryCompleted
18
19  def props(name: String, recovery: Recovery, probe: ActorRef): Props = {
```

test/scala/akka/persistence/AtLeastOnceDeliveryCrashSpec.scala, line 33 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliveryCrashSpec.scala, line 33 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: AtLeastOnceDeliveryCrashSpec$CrashingActor$SendingMessage
File: test/scala/akka/persistence/AtLeastOnceDeliveryCrashSpec.scala:33
Taint Flags:

```
30 case object Message
31 case object CrashMessage
32 case object ConfirmCrashMessage
33 case class SendingMessage(deliveryId: Long)
34 }
35
36 class CrashingActor(testProbe: ActorRef) extends PersistentActor with AtLeastOnceDelivery with ActorLogging {
```

test/scala/akka/persistence/EndToEndEventAdapterSpec.scala, line 27 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EndToEndEventAdapterSpec$B
File: test/scala/akka/persistence/EndToEndEventAdapterSpec.scala:27
Taint Flags:

```
24
25 trait AppModel { def payload: Any }
26 case class A(payload: Any) extends AppModel
27 case class B(payload: Any) extends AppModel
28 case class NewA(payload: Any) extends AppModel
29 case class NewB(payload: Any) extends AppModel
30
```

test/scala/akka/persistence/EventSourcedActorDeleteFailureSpec.scala, line 20 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventSourcedActorDeleteFailureSpec$SimulatedException
File: test/scala/akka/persistence/EventSourcedActorDeleteFailureSpec.scala:20
Taint Flags:

```
17 object EventSourcedActorDeleteFailureSpec {
18
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/EventSourcedActorDeleteFailureSpec.scala, line 20 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
19 case class DeleteTo(n: Long)
20 class SimulatedException(msg: String) extends RuntimeException(msg) with NoStackTrace
21 class SimulatedSerializationException(msg: String) extends RuntimeException(msg) with NoStackTrace
22
23 class DeleteFailingInmemJournal extends InmemJournal {
```

main/scala/akka/persistence/PersistencePlugin.scala, line 25 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistencePlugin$PluginHolder
File: main/scala/akka/persistence/PersistencePlugin.scala:25
Taint Flags:

```
22 */
23 @InternalApi
24 private[akka] object PersistencePlugin {
25 final private[persistence] case class PluginHolder[ScalaDsl, JavaDsl](
26 scaladslPlugin: ScalaDsl,
27 javadslPlugin: JavaDsl)
28 extends Extension
```

test/scala/akka/persistence/EndToEndEventAdapterSpec.scala, line 28 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EndToEndEventAdapterSpec$NewA
File: test/scala/akka/persistence/EndToEndEventAdapterSpec.scala:28
Taint Flags:

```
25 trait AppModel { def payload: Any }
26 case class A(payload: Any) extends AppModel
27 case class B(payload: Any) extends AppModel
28 case class NewA(payload: Any) extends AppModel
29 case class NewB(payload: Any) extends AppModel
30
31 case class JSON(payload: Any)
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence

test/scala/akka/persistence/EndToEndEventAdapterSpec.scala, line 28 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 38 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliveryFailureSpec$Ack
File: test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala:38
Taint Flags:

```
35 case object Start
36 case class Done(ints: Vector[Int])
37
38 case class Ack(i: Int)
39
40 case class Msg(deliveryId: Long, i: Int)
41 case class Confirm(deliveryId: Long, i: Int)
```

test/scala/akka/persistence/ManyRecoveriesSpec.scala, line 21 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ManyRecoveriesSpec$Evt
File: test/scala/akka/persistence/ManyRecoveriesSpec.scala:21
Taint Flags:

```
18 Props(new TestPersistentActor(name, latch))
19
20 final case class Cmd(s: String)
21 final case class Evt(s: String)
22
23 class TestPersistentActor(name: String, latch: Option[TestLatch]) extends PersistentActor {
24
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 37 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 37 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$PostStop
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:37
Taint Flags:

```
34 case class PreStart(name: String)
35 case class PreRestart(name: String)
36 case class PostRestart(name: String)
37 case class PostStop(name: String)
38
39 class A(monitor: ActorRef) extends PersistentActor {
40
```

test/scala/akka/persistence/OptimizedRecoverySpec.scala, line 15 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: OptimizedRecoverySpec$TestPersistentActor$Save
File: test/scala/akka/persistence/OptimizedRecoverySpec.scala:15
Taint Flags:

```
12
13 object TestPersistentActor {
14 case object TakeSnapshot
15 final case class Save(s: String)
16 final case class Saved(s: String, seqNr: Long)
17 case object PersistFromRecoveryCompleted
18
```

main/scala/akka/persistence/AtLeastOnceDelivery.scala, line 61 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

main/scala/akka/persistence/AtLeastOnceDelivery.scala, line 61 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: AtLeastOnceDelivery$UnconfirmedDelivery
File: main/scala/akka/persistence/AtLeastOnceDelivery.scala:61
Taint Flags:

```
58 * Information about a message that has not been confirmed. Included in [[UnconfirmedWarning]]
59 * and [[AtLeastOnceDeliverySnapshot]].
60 */
61 case class UnconfirmedDelivery(deliveryId: Long, destination: ActorPath, message: Any) {
62
63 /**
64 * Java API
```

test/scala/akka/persistence/SnapshotDecodeFailureSpec.scala, line 12 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: SnapshotDecodeFailureSpec$Cmd
File: test/scala/akka/persistence/SnapshotDecodeFailureSpec.scala:12
Taint Flags:

```
9 import akka.testkit.{ EventFilter, ImplicitSender, TestEvent }
10
11 object SnapshotDecodeFailureSpec {
12 case class Cmd(payload: String)
13
14 class SaveSnapshotTestPersistentActor(name: String, probe: ActorRef) extends NamedPersistentActor(name) {
15 def receiveCommand = {
```

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 36 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliveryFailureSpec$Done
File: test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala:36
Taint Flags:

```
33 val numMessages = 10
34
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 36 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
35 case object Start
36 case class Done(ints: Vector[Int])
37
38 case class Ack(i: Int)
39
```

test/scala/akka/persistence/EventAdapterSpec.scala, line 23 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventAdapterSpec$Tagged
File: test/scala/akka/persistence/EventAdapterSpec.scala:23
Taint Flags:

```
20 def payload: Any
21 def tags: immutable.Set[String]
22 }
23 final case class Tagged(payload: Any, tags: immutable.Set[String]) extends JournalModel
24 final case class NotTagged(payload: Any) extends JournalModel {
25 override def tags = Set.empty
26 }
```

test/scala/akka/persistence/RecoveryPermitterSpec.scala, line 19 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: RecoveryPermitterSpec$TestExc
File: test/scala/akka/persistence/RecoveryPermitterSpec.scala:19
Taint Flags:

```
16
17 object RecoveryPermitterSpec {
18
19 class TestExc extends RuntimeException("simulated exc") with NoStackTrace
20
21 def testProps(name: String, probe: ActorRef, throwFromRecoveryCompleted: Boolean = false): Props =
22 Props(new TestPersistentActor(name, probe, throwFromRecoveryCompleted))
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence

test/scala/akka/persistence/RecoveryPermitterSpec.scala, line 19 (Code Correctness: Non-Static Inner Class Implements Serializable) **Low**

main/scala/akka/persistence/AtLeastOnceDelivery.scala, line 72 (Code Correctness: Non-Static Inner Class Implements Serializable) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDelivery$MaxUnconfirmedMessagesExceededException
File: main/scala/akka/persistence/AtLeastOnceDelivery.scala:72
Taint Flags:

```
69 /**
70 * @see [[AtLeastOnceDeliveryLike#maxUnconfirmedMessages]]
71 */
72 class MaxUnconfirmedMessagesExceededException(message: String) extends RuntimeException(message)
73
74 /**
75 * INTERNAL API
```

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 29 (Code Correctness: Non-Static Inner Class Implements Serializable) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliverySpec$Snap
File: test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala:29
Taint Flags:

```
26 case class ActionAck(id: Long)
27 case object Boom
28 case object SaveSnap
29 case class Snap(deliverySnapshot: AtLeastOnceDeliverySnapshot) // typically includes some user data as well
30
31 def senderProps(
32 testActor: ActorRef,
```

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 22 (Code Correctness: Non-Static Inner Class Implements Serializable) **Low**

Issue Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 22 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliverySpec$AcceptedReq
File: test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala:22
Taint Flags:

```
19 case object InvalidReq
20
21 sealed trait Evt
22 case class AcceptedReq(payload: String, destination: ActorPath) extends Evt
23 case class ReqDone(id: Long) extends Evt
24
25 case class Action(id: Long, payload: String)
```

test/scala/akka/persistence/EventSourcedActorDeleteFailureSpec.scala, line 21 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventSourcedActorDeleteFailureSpec$SimulatedSerializationException
File: test/scala/akka/persistence/EventSourcedActorDeleteFailureSpec.scala:21
Taint Flags:

```
18
19 case class DeleteTo(n: Long)
20 class SimulatedException(msg: String) extends RuntimeException(msg) with NoStackTrace
21 class SimulatedSerializationException(msg: String) extends RuntimeException(msg) with NoStackTrace
22
23 class DeleteFailingInmemJournal extends InmemJournal {
24 override def asyncDeleteMessagesTo(persistenceId: String, toSequenceNr: Long): Future[Unit] =
```

main/scala/akka/persistence/AtLeastOnceDelivery.scala, line 46 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

main/scala/akka/persistence/AtLeastOnceDelivery.scala, line 46 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: AtLeastOnceDelivery$UnconfirmedWarning
File: main/scala/akka/persistence/AtLeastOnceDelivery.scala:46
Taint Flags:

```
43  * @see [[AtLeastOnceDeliveryLike#warnAfterNumberOfUnconfirmedAttempts]]
44  */
45  @SerialVersionUID(1L)
46  case class UnconfirmedWarning(unconfirmedDeliveries: immutable.Seq[UnconfirmedDelivery]) {
47
48  /**
49  * Java API
```

test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala, line 24 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: SnapshotFailureRobustnessSpec$DeleteSnapshot
File: test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala:24
Taint Flags:

```
21 object SnapshotFailureRobustnessSpec {
22
23  case class Cmd(payload: String)
24  case class DeleteSnapshot(seqNr: Int)
25  case class DeleteSnapshots(criteria: SnapshotSelectionCriteria)
26
27  class SaveSnapshotTestPersistentActor(name: String, probe: ActorRef) extends NamedPersistentActor(name) {
```

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 44 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliveryFailureSpec$MsgSent
File: test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala:44
Taint Flags:

```
41  case class Confirm(deliveryId: Long, i: Int)
42
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 44 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
43 sealed trait Evt
44 case class MsgSent(i: Int) extends Evt
45 case class MsgConfirmed(deliveryId: Long, i: Int) extends Evt
46
47 trait ChaosSupport { this: Actor =>
```

test/scala/akka/persistence/EventSourcedActorFailureSpec.scala, line 20 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventSourcedActorFailureSpec$SimulatedException
File: test/scala/akka/persistence/EventSourcedActorFailureSpec.scala:20
Taint Flags:

```
17 object EventSourcedActorFailureSpec {
18 import PersistentActorSpec.{ Cmd, Evt, ExamplePersistentActor }
19
20 class SimulatedException(msg: String) extends RuntimeException(msg) with NoStackTrace
21 class SimulatedSerializationException(msg: String) extends RuntimeException(msg) with NoStackTrace
22
23 class FailingInmemJournal extends InmemJournal {
```

test/scala/akka/persistence/TimerPersistentActorSpec.scala, line 24 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TimerPersistentActorSpec$AutoReceivedMessageWrapper
File: test/scala/akka/persistence/TimerPersistentActorSpec.scala:24
Taint Flags:

```
21
22 final case class Scheduled(msg: Any, replyTo: ActorRef)
23
24 final case class AutoReceivedMessageWrapper(msg: AutoReceivedMessage)
25
26 class TestPersistentActor(name: String) extends Timers with PersistentActor {
27
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence

test/scala/akka/persistence/TimerPersistentActorSpec.scala, line 24 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

test/scala/akka/persistence/EventSourcedActorDeleteFailureSpec.scala, line 19 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventSourcedActorDeleteFailureSpec$DeleteTo
File: test/scala/akka/persistence/EventSourcedActorDeleteFailureSpec.scala:19
Taint Flags:

```
16
17 object EventSourcedActorDeleteFailureSpec {
18
19  case class DeleteTo(n: Long)
20  class SimulatedException(msg: String) extends RuntimeException(msg) with NoStackTrace
21  class SimulatedSerializationException(msg: String) extends RuntimeException(msg) with NoStackTrace
22
```

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 23 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliverySpec$ReqDone
File: test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala:23
Taint Flags:

```
20
21  sealed trait Evt
22  case class AcceptedReq(payload: String, destination: ActorPath) extends Evt
23  case class ReqDone(id: Long) extends Evt
24
25  case class Action(id: Long, payload: String)
26  case class ActionAck(id: Long)
```

main/scala/akka/persistence/Eventsourced.scala, line 44 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

main/scala/akka/persistence/Eventsourced.scala, line 44 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Eventsourced$RecoveryTick
File: main/scala/akka/persistence/Eventsourced.scala:44
Taint Flags:

```
41 private[akka] final case class AsyncHandlerInvocation(evt: Any, handler: Any => Unit) extends PendingHandlerInvocation
42
43 /** INTERNAL API: message used to detect that recovery timed out */
44 private[akka] final case class RecoveryTick(snapshot: Boolean)
45 }
46
47 /**
```

test/scala/akka/persistence/TimerPersistentActorSpec.scala, line 22 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TimerPersistentActorSpec$Scheduled
File: test/scala/akka/persistence/TimerPersistentActorSpec.scala:22
Taint Flags:

```
19 def testProps(name: String): Props =
20 Props(new TestPersistentActor(name))
21
22 final case class Scheduled(msg: Any, replyTo: ActorRef)
23
24 final case class AutoReceivedMessageWrapper(msg: AutoReceivedMessage)
25
```

main/scala/akka/persistence/Eventsourced.scala, line 41 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

main/scala/akka/persistence/Eventsourced.scala, line 41 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: Eventsourced$AsyncHandlerInvocation
File: main/scala/akka/persistence/Eventsourced.scala:41
Taint Flags:

```
38 extends PendingHandlerInvocation
39
40 /** INTERNAL API: does not force the actor to stash commands; Originates from either `persistAsync` or `defer` calls */
41 private[akka] final case class AsyncHandlerInvocation(evt: Any, handler: Any => Unit) extends PendingHandlerInvocation
42
43 /** INTERNAL API: message used to detect that recovery timed out */
44 private[akka] final case class RecoveryTick(snapshot: Boolean)
```

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 41 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliveryFailureSpec$Confirm
File: test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala:41
Taint Flags:

```
38 case class Ack(i: Int)
39
40 case class Msg(deliveryId: Long, i: Int)
41 case class Confirm(deliveryId: Long, i: Int)
42
43 sealed trait Evt
44 case class MsgSent(i: Int) extends Evt
```

test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala, line 23 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: SnapshotFailureRobustnessSpec$Cmd
File: test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala:23
Taint Flags:

```
20
21 object SnapshotFailureRobustnessSpec {
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala, line 23 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
22
23 case class Cmd(payload: String)
24 case class DeleteSnapshot(seqNr: Int)
25 case class DeleteSnapshots(criteria: SnapshotSelectionCriteria)
26
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 107 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$Msgs
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:107
Taint Flags:

```
104
105 val journal = JournalPuppet(system).probe
106
107 case class Msgs(msg: Any*)
108
109 def expectWrite(subject: ActorRef, msgs: Msgs*): WriteMessages = {
110 val w = journal.expectMsgType[WriteMessages]
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 31 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$Fail
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:31
Taint Flags:

```
28 case class PersistAsync(id: Int, msgs: Any*) extends Command
29 case class Multi(cmd: Command*) extends Command
30 case class Echo(id: Int) extends Command
31 case class Fail(ex: Throwable) extends Command
32 case class Done(id: Int, sub: Int)
33
34 case class PreStart(name: String)
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 31 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

test/scala/akka/persistence/EndToEndEventAdapterSpec.scala, line 29 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EndToEndEventAdapterSpec$NewB
File: test/scala/akka/persistence/EndToEndEventAdapterSpec.scala:29
Taint Flags:

```
26 case class A(payload: Any) extends AppModel
27 case class B(payload: Any) extends AppModel
28 case class NewA(payload: Any) extends AppModel
29 case class NewB(payload: Any) extends AppModel
30
31 case class JSON(payload: Any)
32
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 30 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorSpec$Delete
File: test/scala/akka/persistence/PersistentActorSpec.scala:30
Taint Flags:

```
27
28 final case class LatchCmd(latch: TestLatch, data: Any) extends NoSerializationVerificationNeeded
29
30 final case class Delete(toSequenceNr: Long)
31
32 abstract class ExamplePersistentActor(name: String) extends NamedPersistentActor(name) {
33 var events: List[Any] = Nil
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 34 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 34 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$PreStart
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:34
Taint Flags:

```
31 case class Fail(ex: Throwable) extends Command
32 case class Done(id: Int, sub: Int)
33
34 case class PreStart(name: String)
35 case class PreRestart(name: String)
36 case class PostRestart(name: String)
37 case class PostStop(name: String)
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 32 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$Done
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:32
Taint Flags:

```
29 case class Multi(cmd: Command*) extends Command
30 case class Echo(id: Int) extends Command
31 case class Fail(ex: Throwable) extends Command
32 case class Done(id: Int, sub: Int)
33
34 case class PreStart(name: String)
35 case class PreRestart(name: String)
```

test/scala/akka/persistence/SnapshotSpec.scala, line 73 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotSpec.scala, line 73 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: SnapshotSpec$DeleteN
File: test/scala/akka/persistence/SnapshotSpec.scala:73
Taint Flags:

```
70 }
71
72 final case class Delete1(metadata: SnapshotMetadata)
73 final case class DeleteN(criteria: SnapshotSelectionCriteria)
74
75 class DeleteSnapshotTestPersistentActor(name: String, _recovery: Recovery, probe: ActorRef)
76 extends LoadSnapshotTestPersistentActor(name, _recovery, probe) {
```

main/scala/akka/persistence/Eventsourced.scala, line 37 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Eventsourced$StashingHandlerInvocation
File: main/scala/akka/persistence/Eventsourced.scala:37
Taint Flags:

```
34 }
35
36 /** INTERNAL API: forces actor to stash incoming commands until all these invocations are handled */
37 private[akka] final case class StashingHandlerInvocation(evt: Any, handler: Any => Unit)
38 extends PendingHandlerInvocation
39
40 /** INTERNAL API: does not force the actor to stash commands; Originates from either `persistAsync` or `defer` calls */
```

test/scala/akka/persistence/ManyRecoveriesSpec.scala, line 20 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ManyRecoveriesSpec$Cmd
File: test/scala/akka/persistence/ManyRecoveriesSpec.scala:20
Taint Flags:

```
17 def testProps(name: String, latch: Option[TestLatch]): Props =
18 Props(new TestPersistentActor(name, latch))
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/ManyRecoveriesSpec.scala, line 20 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
19
20 final case class Cmd(s: String)
21 final case class Evt(s: String)
22
23 class TestPersistentActor(name: String, latch: Option[TestLatch]) extends PersistentActor {
```

test/scala/akka/persistence/EventSourcedActorFailureSpec.scala, line 21 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventSourcedActorFailureSpec$SimulatedSerializationException
File: test/scala/akka/persistence/EventSourcedActorFailureSpec.scala:21
Taint Flags:

```
18 import PersistentActorSpec.{ Cmd, Evt, ExamplePersistentActor }
19
20 class SimulatedException(msg: String) extends RuntimeException(msg) with NoStackTrace
21 class SimulatedSerializationException(msg: String) extends RuntimeException(msg) with NoStackTrace
22
23 class FailingInmemJournal extends InmemJournal {
24
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 28 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorSpec$LatchCmd
File: test/scala/akka/persistence/PersistentActorSpec.scala:28
Taint Flags:

```
25
26 final case class Evt(data: Any)
27
28 final case class LatchCmd(latch: TestLatch, data: Any) extends NoSerializationVerificationNeeded
29
30 final case class Delete(toSequenceNr: Long)
31
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorSpec.scala, line 28 (Code Correctness: Non-Static Inner Class Implements Serializable) — **Low**

main/scala/akka/persistence/AtLeastOnceDelivery.scala, line 27 (Code Correctness: Non-Static Inner Class Implements Serializable) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDelivery$AtLeastOnceDeliverySnapshot
File: main/scala/akka/persistence/AtLeastOnceDelivery.scala:27
Taint Flags:

```
24 * with [[AtLeastOnceDeliveryLike#setDeliverySnapshot]].
25 */
26 @SerialVersionUID(1L)
27 case class AtLeastOnceDeliverySnapshot(
28 currentDeliveryId: Long,
29 unconfirmedDeliveries: immutable.Seq[UnconfirmedDelivery])
30 extends Message {
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 36 (Code Correctness: Non-Static Inner Class Implements Serializable) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$PostRestart
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:36
Taint Flags:

```
33
34 case class PreStart(name: String)
35 case class PreRestart(name: String)
36 case class PostRestart(name: String)
37 case class PostStop(name: String)
38
39 class A(monitor: ActorRef) extends PersistentActor {
```

test/scala/akka/persistence/SnapshotSpec.scala, line 72 (Code Correctness: Non-Static Inner Class Implements Serializable) — **Low**

Issue Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotSpec.scala, line 72 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: SnapshotSpec$Delete1
File: test/scala/akka/persistence/SnapshotSpec.scala:72
Taint Flags:

```
69 }
70 }
71
72 final case class Delete1(metadata: SnapshotMetadata)
73 final case class DeleteN(criteria: SnapshotSelectionCriteria)
74
75 class DeleteSnapshotTestPersistentActor(name: String, _recovery: Recovery, probe: ActorRef)
```

test/scala/akka/persistence/LoadPluginSpec.scala, line 25 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LoadPluginSpec$JournalWithStartupNotification$Started
File: test/scala/akka/persistence/LoadPluginSpec.scala:25
Taint Flags:

```
22 }
23
24 object JournalWithStartupNotification {
25 final case class Started(configPath: String)
26 }
27 class JournalWithStartupNotification(@unused config: Config, configPath: String) extends InmemJournal {
28 context.system.eventStream.publish(JournalWithStartupNotification.Started(configPath))
```

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 25 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 25 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: AtLeastOnceDeliverySpec$Action
File: test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala:25
Taint Flags:

```
22 case class AcceptedReq(payload: String, destination: ActorPath) extends Evt
23 case class ReqDone(id: Long) extends Evt
24
25 case class Action(id: Long, payload: String)
26 case class ActionAck(id: Long)
27 case object Boom
28 case object SaveSnap
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 27 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$Persist
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:27
Taint Flags:

```
24 """)
25
26 sealed trait Command
27 case class Persist(id: Int, msgs: Any*) extends Command
28 case class PersistAsync(id: Int, msgs: Any*) extends Command
29 case class Multi(cmd: Command*) extends Command
30 case class Echo(id: Int) extends Command
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 30 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$Echo
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:30
Taint Flags:

```
27 case class Persist(id: Int, msgs: Any*) extends Command
28 case class PersistAsync(id: Int, msgs: Any*) extends Command
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 30 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
29 case class Multi(cmd: Command*) extends Command
30 case class Echo(id: Int) extends Command
31 case class Fail(ex: Throwable) extends Command
32 case class Done(id: Int, sub: Int)
33
```

test/scala/akka/persistence/EventAdapterSpec.scala, line 31 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventAdapterSpec$UserDataChanged
File: test/scala/akka/persistence/EventAdapterSpec.scala:31
Taint Flags:

```
28 final val DomainEventClassName = classOf[EventAdapterSpec].getCanonicalName + "$" + classOf[DomainEvent].getSimpleName
29 trait DomainEvent
30 final case class TaggedDataChanged(tags: immutable.Set[String], value: Int) extends DomainEvent
31 final case class UserDataChanged(countryCode: String, age: Int) extends DomainEvent
32
33 class UserAgeTaggingAdapter extends EventAdapter {
34 val Adult = Set("adult")
```

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 26 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliverySpec$ActionAck
File: test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala:26
Taint Flags:

```
23 case class ReqDone(id: Long) extends Evt
24
25 case class Action(id: Long, payload: String)
26 case class ActionAck(id: Long)
27 case object Boom
28 case object SaveSnap
29 case class Snap(deliverySnapshot: AtLeastOnceDeliverySnapshot) // typically includes some user data as well
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 26 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 28 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$PersistAsync
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:28
Taint Flags:

```
25
26 sealed trait Command
27 case class Persist(id: Int, msgs: Any*) extends Command
28 case class PersistAsync(id: Int, msgs: Any*) extends Command
29 case class Multi(cmd: Command*) extends Command
30 case class Echo(id: Int) extends Command
31 case class Fail(ex: Throwable) extends Command
```

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 45 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliveryFailureSpec$MsgConfirmed
File: test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala:45
Taint Flags:

```
42
43 sealed trait Evt
44 case class MsgSent(i: Int) extends Evt
45 case class MsgConfirmed(deliveryId: Long, i: Int) extends Evt
46
47 trait ChaosSupport { this: Actor =>
48 def random = ThreadLocalRandom.current
```

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 35 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 35 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$PreRestart
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:35
Taint Flags:

```
32 case class Done(id: Int, sub: Int)
33
34 case class PreStart(name: String)
35 case class PreRestart(name: String)
36 case class PostRestart(name: String)
37 case class PostStop(name: String)
38
```

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 17 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AtLeastOnceDeliverySpec$Req
File: test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala:17
Taint Flags:

```
14
15 object AtLeastOnceDeliverySpec {
16
17 case class Req(payload: String)
18 case object ReqAck
19 case object InvalidReq
20
```

test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala, line 25 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala, line 25 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: SnapshotFailureRobustnessSpec$DeleteSnapshots
File: test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala:25
Taint Flags:

```
22
23 case class Cmd(payload: String)
24 case class DeleteSnapshot(seqNr: Int)
25 case class DeleteSnapshots(criteria: SnapshotSelectionCriteria)
26
27 class SaveSnapshotTestPersistentActor(name: String, probe: ActorRef) extends NamedPersistentActor(name) {
28 override def receiveRecover: Receive = {
```

test/scala/akka/persistence/EventAdapterSpec.scala, line 24 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventAdapterSpec$NotTagged
File: test/scala/akka/persistence/EventAdapterSpec.scala:24
Taint Flags:

```
21 def tags: immutable.Set[String]
22 }
23 final case class Tagged(payload: Any, tags: immutable.Set[String]) extends JournalModel
24 final case class NotTagged(payload: Any) extends JournalModel {
25 override def tags = Set.empty
26 }
27
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 24 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorSpec$Cmd
File: test/scala/akka/persistence/PersistentActorSpec.scala:24
Taint Flags:

```
21
22 object PersistentActorSpec {
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorSpec.scala, line 24 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
23
24  final case class Cmd(data: Any)
25
26  final case class Evt(data: Any)
27
```

test/scala/akka/persistence/EndToEndEventAdapterSpec.scala, line 31 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EndToEndEventAdapterSpec$JSON
File: test/scala/akka/persistence/EndToEndEventAdapterSpec.scala:31
Taint Flags:

```
28  case class NewA(payload: Any) extends AppModel
29  case class NewB(payload: Any) extends AppModel
30
31  case class JSON(payload: Any)
32
33  class AEndToEndAdapter(@unused system: ExtendedActorSystem) extends EventAdapter {
34  override def manifest(event: Any): String = event.getClass.getCanonicalName
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 19 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorStashingSpec$Cmd
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:19
Taint Flags:

```
16 import akka.util.unused
17
18 object PersistentActorStashingSpec {
19  final case class Cmd(data: Any)
20  final case class Evt(data: Any)
21
22  abstract class StashExamplePersistentActor(name: String) extends NamedPersistentActor(name) {
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 19 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala, line 29 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentActorJournalProtocolSpec$Multi
File: test/scala/akka/persistence/PersistentActorJournalProtocolSpec.scala:29
Taint Flags:

```
26 sealed trait Command
27 case class Persist(id: Int, msgs: Any*) extends Command
28 case class PersistAsync(id: Int, msgs: Any*) extends Command
29 case class Multi(cmd: Command*) extends Command
30 case class Echo(id: Int) extends Command
31 case class Fail(ex: Throwable) extends Command
32 case class Done(id: Int, sub: Int)
```

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 303 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$TimeoutMarker
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:303
Taint Flags:

```
300
301 /** INTERNAL API */
302 @InternalApi
303 private[persistence] final case class TimeoutMarker(generation: Long)
304
305 /** INTERNAL API */
306 @InternalApi
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 478 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$Event
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:478
Taint Flags:

```
475 * All messages sent to the [[akka.actor.FSM]] will be wrapped inside an
476 * `Event`, which allows pattern matching to extract both state and data.
477 */
478 final case class Event[D](event: Any, stateData: D) extends NoSerializationVerificationNeeded
479
480 /**
481 * Case class representing the state of the [[akka.actor.FSM]] whithin the
```

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 584 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSMSpec$TimeoutFSM$State
File: test/scala/akka/persistence/fsm/PersistentFSMSpec.scala:584
Taint Flags:

```
581
582 object TimeoutFSM {
583 val OverrideTimeoutToInf = "override-timeout-to-inf"
584 case class State(identifier: String) extends PersistentFSM.FSMState
585 def props(probe: ActorRef) = Props(new TimeoutFSM(probe))
586 }
587
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 210 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 210 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: PersistentFSM$StateChangeEvent
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:210
Taint Flags:

```
207 * @param stateIdentifier FSM state identifier
208 * @param timeout FSM state timeout
209 */
210 case class StateChangeEvent(stateIdentifier: String, timeout: Option[FiniteDuration]) extends PersistentFsmEvent
211
212 /**
213 * FSM state and data snapshot
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 378 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$State
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:378
Taint Flags:

```
375 * accumulated while processing the last message, possibly domain event and handler
376 * to be executed after FSM moves to the new state (also triggered when staying in the same state)
377 */
378 final case class State[S, D, E](
379 stateName: S,
380 stateData: D,
381 timeout: Option[FiniteDuration] = None,
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 258 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$Transition
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:258
Taint Flags:

```
255 * Message type which is used to communicate transitions between states to
256 * all subscribed listeners (use [[akka.actor.FSM.SubscribeTransitionCallBack]]).
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 258 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
257 */
258 final case class Transition[S](fsmRef: ActorRef, from: S, to: S, timeout: Option[FiniteDuration])
259
260 /**
261 * Send this to an [[akka.actor.FSM]] to request first the [[PersistentFSM.CurrentState]]
```

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 430 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSMSpec$NonEmptyShoppingCart
File: test/scala/akka/persistence/fsm/PersistentFSMSpec.scala:430
Taint Flags:

```
427 def addItem(item: Item) = NonEmptyShoppingCart(item :: Nil)
428 def empty() = this
429 }
430 case class NonEmptyShoppingCart(items: Seq[Item]) extends ShoppingCart {
431 def addItem(item: Item) = NonEmptyShoppingCart(items :+ item)
432 def empty() = EmptyShoppingCart
433 }
```

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 420 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSMSpec$Item
File: test/scala/akka/persistence/fsm/PersistentFSMSpec.scala:420
Taint Flags:

```
417 //#customer-states
418
419 //#customer-states-data
420 case class Item(id: String, name: String, price: Float)
421
422 sealed trait ShoppingCart {
423 def addItem(item: Item): ShoppingCart
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence.fsm

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 420 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 454 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSMSpec$PurchaseWasMade
File: test/scala/akka/persistence/fsm/PersistentFSMSpec.scala:454
Taint Flags:

```
451
452 //Side effects - report events to be sent to some "Report Actor"
453 sealed trait ReportEvent
454 case class PurchaseWasMade(items: Seq[Item]) extends ReportEvent
455 case object ShoppingCardDiscarded extends ReportEvent
456
457 class SimpleTransitionFSM(_persistenceId: String, reportActor: ActorRef)(
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 221 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$PersistentFSMSnapshot
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:221
Taint Flags:

```
218 * @tparam D state data type
219 */
220 @InternalApi
221 private[persistence] case class PersistentFSMSnapshot[D](
222 stateIdentifier: String,
223 data: D,
224 timeout: Option[FiniteDuration])
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 265 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 265 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$SubscribeTransitionCallBack
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:265
Taint Flags:

```
262 * and then a series of [[PersistentFSM.Transition]] updates. Cancel the subscription
263 * using [[PersistentFSM.UnsubscribeTransitionCallBack]].
264 */
265 final case class SubscribeTransitionCallBack(actorRef: ActorRef)
266
267 /**
268 * Unsubscribe from [[akka.actor.FSM.Transition]] notifications which was
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 294 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$Failure
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:294
Taint Flags:

```
291 * an error, e.g. if the state to transition into does not exist. You can use
292 * this to communicate a more precise cause to the `onTermination` block.
293 */
294 final case class Failure(cause: Any) extends Reason
295
296 /**
297 * This case object is received in case of a state timeout.
```

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 438 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.fsm

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 438 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: PersistentFSMSpec$AddItem
File: test/scala/akka/persistence/fsm/PersistentFSMSpec.scala:438
Taint Flags:

```
435
436 //#customer-commands
437 sealed trait Command
438 case class AddItem(item: Item) extends Command
439 case object Buy extends Command
440 case object Leave extends Command
441 case object GetCurrentCart extends Command
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 271 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$UnsubscribeTransitionCallBack
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:271
Taint Flags:

```
268 * Unsubscribe from [[akka.actor.FSM.Transition]] notifications which was
269 * effected by sending the corresponding [[akka.actor.FSM.SubscribeTransitionCallBack]].
270 */
271 final case class UnsubscribeTransitionCallBack(actorRef: ActorRef)
272
273 /**
274 * Reason why this [[akka.actor.FSM]] is shutting down.
```

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 614 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSMSpec$IntAdded
File: test/scala/akka/persistence/fsm/PersistentFSMSpec.scala:614
Taint Flags:

```
611 case object Persist4xAtOnce extends SnapshotFSMState { override def identifier: String = "Persist4xAtOnce" }
612
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.fsm

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 614 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
613 sealed trait SnapshotFSMEvent
614 case class IntAdded(i: Int) extends SnapshotFSMEvent
615
616 object SnapshotFSM {
617 def props(probe: ActorRef) = Props(new SnapshotFSM(probe))
```

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 446 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSMSpec$ItemAdded
File: test/scala/akka/persistence/fsm/PersistentFSMSpec.scala:446
Taint Flags:

```
443
444 //#customer-domain-events
445 sealed trait DomainEvent
446 case class ItemAdded(item: Item) extends DomainEvent
447 case object OrderExecuted extends DomainEvent
448 case object OrderDiscarded extends DomainEvent
449 case object CustomerInactive extends DomainEvent
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 370 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$LogEntry
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:370
Taint Flags:

```
367 /**
368 * Log Entry of the [[akka.actor.LoggingFSM]], can be obtained by calling `getLog`.
369 */
370 final case class LogEntry[S, D](stateName: S, stateData: D, event: Any)
371
372 /**
373 * This captures all of the managed state of the [[akka.actor.FSM]]: the state
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 370 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 252 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$CurrentState
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:252
Taint Flags:

```
249 * [[akka.actor.FSM.SubscribeTransitionCallBack]] before sending any
250 * [[akka.actor.FSM.Transition]] messages.
251 */
252 final case class CurrentState[S](fsmRef: ActorRef, state: S, timeout: Option[FiniteDuration])
253
254 /**
255 * Message type which is used to communicate transitions between states to
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 484 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$StopEvent
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:484
Taint Flags:

```
481 * Case class representing the state of the [[akka.actor.FSM]] whithin the
482 * `onTermination` block.
483 */
484 final case class StopEvent[S, D](reason: Reason, currentState: S, stateData: D)
485 extends NoSerializationVerificationNeeded
486
487 }
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 333 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 333 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistentFSM$Timer
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:333
Taint Flags:

```
330 * INTERNAL API
331 */
332 @InternalApi
333 private[persistence] final case class Timer(name: String, msg: Any, mode: TimerMode, generation: Int, owner: AnyRef)(
334 context: ActorContext)
335 extends NoSerializationVerificationNeeded {
336 private var ref: Option[Cancellable] = _
```

Package: akka.persistence.journal

main/scala/akka/persistence/journal/PersistencePluginProxy.scala, line 29 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PersistencePluginProxy$TargetLocation
File: main/scala/akka/persistence/journal/PersistencePluginProxy.scala:29
Taint Flags:

```
26 import akka.util.Helpers.Requiring
27
28 object PersistencePluginProxy {
29 final case class TargetLocation(address: Address)
30 private case object InitTimeout
31
32 def setTargetLocation(system: ActorSystem, address: Address): Unit = {
```

main/scala/akka/persistence/journal/AsyncWriteProxy.scala, line 101 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.journal

main/scala/akka/persistence/journal/AsyncWriteProxy.scala, line 101 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: AsyncWriteProxy$SetStore
File: main/scala/akka/persistence/journal/AsyncWriteProxy.scala:101
Taint Flags:

```
98  * INTERNAL API.
99  */
100 private[persistence] object AsyncWriteProxy {
101  final case class SetStore(ref: ActorRef)
102  case object InitTimeout
103 }
104
```

main/scala/akka/persistence/journal/AsyncWriteJournal.scala, line 299 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AsyncWriteJournal$Desequenced
File: main/scala/akka/persistence/journal/AsyncWriteJournal.scala:299
Taint Flags:

```
296 private[persistence] object AsyncWriteJournal {
297  val successUnit: Success[Unit] = Success(())
298
299  final case class Desequenced(msg: Any, snr: Long, target: ActorRef, sender: ActorRef)
300  extends NoSerializationVerificationNeeded
301
302  class Resequencer extends Actor {
```

main/scala/akka/persistence/journal/EventAdapters.scala, line 129 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventAdapters$CombinedReadEventAdapter
File: main/scala/akka/persistence/journal/EventAdapters.scala:129
Taint Flags:

```
126  }
127
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.journal

main/scala/akka/persistence/journal/EventAdapters.scala, line 129 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
128 /** INTERNAL API */
129 private[akka] case class CombinedReadEventAdapter(adapters: immutable.Seq[EventAdapter]) extends EventAdapter {
130 private def onlyReadSideException =
131 new IllegalStateException("CombinedReadEventAdapter must not be used when writing (creating manifests) events!")
132 override def manifest(event: Any): String = throw onlyReadSideException
```

Package: akka.persistence.journal.inmem

test/scala/akka/persistence/journal/inmem/InmemJournalSpec.scala, line 17 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: InmemJournalSpec$Cmd
File: test/scala/akka/persistence/journal/inmem/InmemJournalSpec.scala:17
Taint Flags:

```
14 def testProps(name: String): Props =
15 Props(new TestPersistentActor(name))
16
17 final case class Cmd(s: String)
18 final case class Delete(toSeqNr: Long)
19 final case class Evt(s: String)
20
```

test/scala/akka/persistence/journal/inmem/InmemJournalSpec.scala, line 19 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: InmemJournalSpec$Evt
File: test/scala/akka/persistence/journal/inmem/InmemJournalSpec.scala:19
Taint Flags:

```
16
17 final case class Cmd(s: String)
18 final case class Delete(toSeqNr: Long)
19 final case class Evt(s: String)
20
21 class TestPersistentActor(name: String) extends PersistentActor {
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence.journal.inmem

test/scala/akka/persistence/journal/inmem/InmemJournalSpec.scala, line 19 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

22

main/scala/akka/persistence/journal/inmem/InmemJournal.scala, line 38 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: InmemJournal$Write
File: main/scala/akka/persistence/journal/inmem/InmemJournal.scala:38
Taint Flags:

```
35 object InmemJournal {
36  sealed trait Operation
37
38  final case class Write(event: Any, persistenceId: String, sequenceNr: Long) extends Operation
39  final case class Delete(persistenceId: String, toSequenceNr: Long) extends Operation
40
41  @InternalApi
```

main/scala/akka/persistence/journal/inmem/InmemJournal.scala, line 42 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: InmemJournal$ReplayWithMeta
File: main/scala/akka/persistence/journal/inmem/InmemJournal.scala:42
Taint Flags:

```
39  final case class Delete(persistenceId: String, toSequenceNr: Long) extends Operation
40
41  @InternalApi
42  private[persistence] case class ReplayWithMeta(
43  from: Long,
44  to: Long,
45  limit: Long,
```

main/scala/akka/persistence/journal/inmem/InmemJournal.scala, line 39 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.journal.inmem

main/scala/akka/persistence/journal/inmem/InmemJournal.scala, line 39 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: InmemJournal$Delete
File: main/scala/akka/persistence/journal/inmem/InmemJournal.scala:39
Taint Flags:

```
36 sealed trait Operation
37
38 final case class Write(event: Any, persistenceId: String, sequenceNr: Long) extends Operation
39 final case class Delete(persistenceId: String, toSequenceNr: Long) extends Operation
40
41 @InternalApi
42 private[persistence] case class ReplayWithMeta(
```

main/scala/akka/persistence/journal/inmem/InmemJournal.scala, line 49 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: InmemJournal$MessageWithMeta
File: main/scala/akka/persistence/journal/inmem/InmemJournal.scala:49
Taint Flags:

```
46 persistenceId: String,
47 replyTo: ActorRef)
48 @InternalApi
49 private[persistence] case class MessageWithMeta(pr: PersistentRepr, meta: OptionVal[Any])
50 }
51
52 /**
```

test/scala/akka/persistence/journal/inmem/InmemJournalSpec.scala, line 18 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence.journal.inmem

test/scala/akka/persistence/journal/inmem/InmemJournalSpec.scala, line 18 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: InmemJournalSpec$Delete
File: test/scala/akka/persistence/journal/inmem/InmemJournalSpec.scala:18
Taint Flags:

```
15 Props(new TestPersistentActor(name))
16
17 final case class Cmd(s: String)
18 final case class Delete(toSeqNr: Long)
19 final case class Evt(s: String)
20
21 class TestPersistentActor(name: String) extends PersistentActor {
```

Package: akka.persistence.journal.leveldb

test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala, line 179 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: JournalCompactionSpec$EventLogger$Event
File: test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala:179
Taint Flags:

```
176
177 case class Delete(toSeqNr: Long)
178
179 case class Event(seqNr: Long, payload: String)
180
181 def props(specId: String, watcher: ActorRef): Props = Props(classOf[EventLogger], specId, watcher)
182 }
```

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 109 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeveldbJournal$TaggedEventAppended
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:109
Taint Flags:

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.journal.leveldb

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 109 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
106 * via an [[akka.persistence.journal.EventAdapter]].
107 */
108 final case class SubscribeTag(tag: String) extends SubscriptionCommand
109 final case class TaggedEventAppended(tag: String) extends DeadLetterSuppression
110
111 /**
112 * `fromSequenceNr` is exclusive
```

test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala, line 175 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: JournalCompactionSpec$EventLogger$Generated
File: test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala:175
Taint Flags:

```
172
173 case object Generate
174
175 case class Generated(seqNr: Long)
176
177 case class Delete(toSeqNr: Long)
178
```

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 122 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeveldbJournal$ReplayedTaggedMessage
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:122
Taint Flags:

```
119 tag: String,
120 replyTo: ActorRef)
121 extends SubscriptionCommand
122 final case class ReplayedTaggedMessage(persistent: PersistentRepr, tag: String, offset: Long)
123 extends DeadLetterSuppression
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.persistence.journal.leveldb

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 122 (Code Correctness: Non-Static Inner Class Implements Serializable) **Low**

```
124 with NoSerializationVerificationNeeded
125 }
```

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 88 (Code Correctness: Non-Static Inner Class Implements Serializable) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeveldbJournal$SubscribePersistenceId
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:88
Taint Flags:

```
85 * Used by query-side. The journal will send [[EventAppended]] messages to
86 * the subscriber when `asyncWriteMessages` has been called.
87 */
88 final case class SubscribePersistenceId(persistenceId: String) extends SubscriptionCommand
89 final case class EventAppended(persistenceId: String) extends DeadLetterSuppression
90
91 /**
```

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 115 (Code Correctness: Non-Static Inner Class Implements Serializable) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeveldbJournal$ReplayTaggedMessages
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:115
Taint Flags:

```
112 * `fromSequenceNr` is exclusive
113 * `toSequenceNr` is inclusive
114 */
115 final case class ReplayTaggedMessages(
116 fromSequenceNr: Long,
117 toSequenceNr: Long,
118 max: Long,
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.journal.leveldb

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 99 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeveldbJournal$PersistenceIdAdded
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:99
Taint Flags:

```
96  */
97  case object SubscribeAllPersistenceIds extends SubscriptionCommand
98  final case class CurrentPersistenceIds(allPersistenceIds: Set[String]) extends DeadLetterSuppression
99  final case class PersistenceIdAdded(persistenceId: String) extends DeadLetterSuppression
100
101 /**
102 * Subscribe the `sender` to changes (appended events) for a specific `tag`.
```

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 98 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeveldbJournal$CurrentPersistenceIds
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:98
Taint Flags:

```
95  * are created.
96  */
97  case object SubscribeAllPersistenceIds extends SubscriptionCommand
98  final case class CurrentPersistenceIds(allPersistenceIds: Set[String]) extends DeadLetterSuppression
99  final case class PersistenceIdAdded(persistenceId: String) extends DeadLetterSuppression
100
101 /**
```

test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala, line 177 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.journal.leveldb

test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala, line 177 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: JournalCompactionSpec$EventLogger$Delete
File: test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala:177
Taint Flags:

```
174
175 case class Generated(seqNr: Long)
176
177 case class Delete(toSeqNr: Long)
178
179 case class Event(seqNr: Long, payload: String)
180
```

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 89 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeveldbJournal$EventAppended
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:89
Taint Flags:

```
86 * the subscriber when `asyncWriteMessages` has been called.
87 */
88 final case class SubscribePersistenceId(persistenceId: String) extends SubscriptionCommand
89 final case class EventAppended(persistenceId: String) extends DeadLetterSuppression
90
91 /**
92 * Subscribe the `sender` to current and new persistenceIds.
```

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 108 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeveldbJournal$SubscribeTag
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:108
Taint Flags:

```
105 * Events are tagged by wrapping in [[akka.persistence.journal.Tagged]]
106 * via an [[akka.persistence.journal.EventAdapter]].
```

Code Correctness: Non-Static Inner Class Implements Serializable — Low

Package: akka.persistence.journal.leveldb

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 108 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
107 */
108 final case class SubscribeTag(tag: String) extends SubscriptionCommand
109 final case class TaggedEventAppended(tag: String) extends DeadLetterSuppression
110
111 /**
```

main/scala/akka/persistence/journal/leveldb/LeveldbCompaction.scala, line 11 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeveldbCompaction$TryCompactLeveldb
File: main/scala/akka/persistence/journal/leveldb/LeveldbCompaction.scala:11
Taint Flags:

```
8
9 private[persistence] object LeveldbCompaction {
10
11 case class TryCompactLeveldb(persistenceId: String, toSeqNr: Long)
12 }
13
14 /**
```

Package: akka.persistence.snapshot

main/scala/akka/persistence/snapshot/NoSnapshotStore.scala, line 20 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: NoSnapshotStore$NoSnapshotStoreException
File: main/scala/akka/persistence/snapshot/NoSnapshotStore.scala:20
Taint Flags:

```
17 */
18 final class NoSnapshotStore extends SnapshotStore {
19
20 final class NoSnapshotStoreException extends RuntimeException("No snapshot store configured!")
21
22 private val flop: Future[Nothing] =
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low
Package: akka.persistence.snapshot	
main/scala/akka/persistence/snapshot/NoSnapshotStore.scala, line 20 (Code Correctness: Non-Static Inner Class Implements Serializable)	**Low**

```
23  Future.failed(new NoSnapshotStoreException)
```

Code Correctness: Non-Synchronized Method Overrides Synchronized Method (6 issues)

Abstract

Synchronized methods should not be overridden with non-syncrhonized methods.

Explanation

A parent class declared the method `synchronized`, guaranteeing correct behavior when multiple threads access the same instance. All overriding methods should also be declared `synchronized`, otherwise unexpected behavior may occur. **Example 1:** In the following code, the class `Foo` overrides the class `Bar` but does not declare the method `synchronizedMethod` to be `synchronized`:

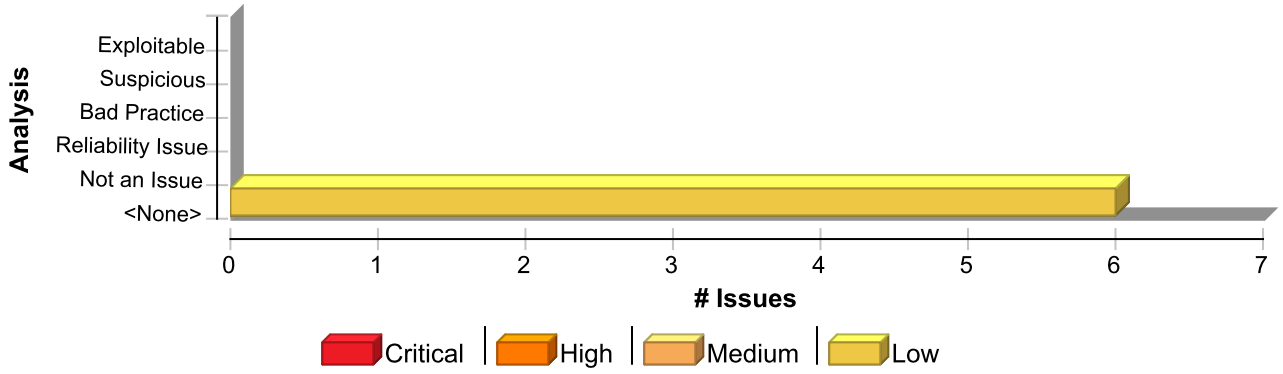
```
public class Bar {
public synchronized void synchronizedMethod() {
    for (int i=0; i<10; i++) System.out.print(i);
    System.out.println();
}
}

public class Foo extends Bar {
public void synchronizedMethod() {
    for (int i=0; i<10; i++) System.out.print(i);
    System.out.println();
}
}
```

In this case, an instance of `Foo` could be cast to type `Bar`. If the same instance is given to two separate threads and `synchronizedMethod` is executed repeatedly, the behavior will be unpredictable.

Recommendation

If the parent method is `synchronized`, the method must be declared `synchronized`.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Non-Synchronized Method Overrides Synchronized Method	6	0	0	6
Total	**6**	**0**	**0**	**6**

Code Correctness: Non-Synchronized Method Overrides Synchronized Method — Low

Package: akka.persistence.journal.leveldb

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 28 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: numericId
Enclosing Method: numericId()
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:28
Taint Flags:

```
25  * Journal backed by a local LevelDB store. For production use.
26  */
27 @deprecated("Use another journal implementation", "2.6.15")
28 private[persistence] class LeveldbJournal(cfg: Config) extends AsyncWriteJournal with LeveldbStore {
29   import LeveldbJournal._
30
31   def this() = this(LeveldbStore.emptyConfig)
```

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 28 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: allPersistenceIds
Enclosing Method: allPersistenceIds()
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:28
Taint Flags:

```
25  * Journal backed by a local LevelDB store. For production use.
26  */
27 @deprecated("Use another journal implementation", "2.6.15")
28 private[persistence] class LeveldbJournal(cfg: Config) extends AsyncWriteJournal with LeveldbStore {
29   import LeveldbJournal._
30
31   def this() = this(LeveldbStore.emptyConfig)
```

main/scala/akka/persistence/journal/leveldb/SharedLeveldbStore.scala, line 25 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Non-Synchronized Method Overrides Synchronized Method	Low

Package: akka.persistence.journal.leveldb

main/scala/akka/persistence/journal/leveldb/SharedLeveldbStore.scala, line 25 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method)	Low

Sink Details

Sink: Function: isNewPersistenceId
Enclosing Method: isNewPersistenceId()
File: main/scala/akka/persistence/journal/leveldb/SharedLeveldbStore.scala:25
Taint Flags:

```
22  * shared LevelDB store is for testing only.
23  */
24 @deprecated("Use another journal implementation or the in-mem journal in combination with the journal-proxy", "2.6.15")
25 class SharedLeveldbStore(cfg: Config) extends LeveldbStore {
26  import AsyncWriteTarget._
27  import context.dispatcher
28
```

main/scala/akka/persistence/journal/leveldb/SharedLeveldbStore.scala, line 25 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: numericId
Enclosing Method: numericId()
File: main/scala/akka/persistence/journal/leveldb/SharedLeveldbStore.scala:25
Taint Flags:

```
22  * shared LevelDB store is for testing only.
23  */
24 @deprecated("Use another journal implementation or the in-mem journal in combination with the journal-proxy", "2.6.15")
25 class SharedLeveldbStore(cfg: Config) extends LeveldbStore {
26  import AsyncWriteTarget._
27  import context.dispatcher
28
```

main/scala/akka/persistence/journal/leveldb/SharedLeveldbStore.scala, line 25 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: allPersistenceIds
Enclosing Method: allPersistenceIds()

Code Correctness: Non-Synchronized Method Overrides Synchronized Method	Low

Package: akka.persistence.journal.leveldb

main/scala/akka/persistence/journal/leveldb/SharedLeveldbStore.scala, line 25 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method)	Low

File: main/scala/akka/persistence/journal/leveldb/SharedLeveldbStore.scala:25
Taint Flags:

```
22  * shared LevelDB store is for testing only.
23  */
24 @deprecated("Use another journal implementation or the in-mem journal in combination with the journal-proxy", "2.6.15")
25 class SharedLeveldbStore(cfg: Config) extends LeveldbStore {
26  import AsyncWriteTarget._
27  import context.dispatcher
28
```

main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala, line 28 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: isNewPersistenceId
Enclosing Method: isNewPersistenceId()
File: main/scala/akka/persistence/journal/leveldb/LeveldbJournal.scala:28
Taint Flags:

```
25  * Journal backed by a local LevelDB store. For production use.
26  */
27 @deprecated("Use another journal implementation", "2.6.15")
28 private[persistence] class LeveldbJournal(cfg: Config) extends AsyncWriteJournal with LeveldbStore {
29  import LeveldbJournal._
30
31  def this() = this(LeveldbStore.emptyConfig)
```

Dead Code: Expression is Always false (40 issues)

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`.

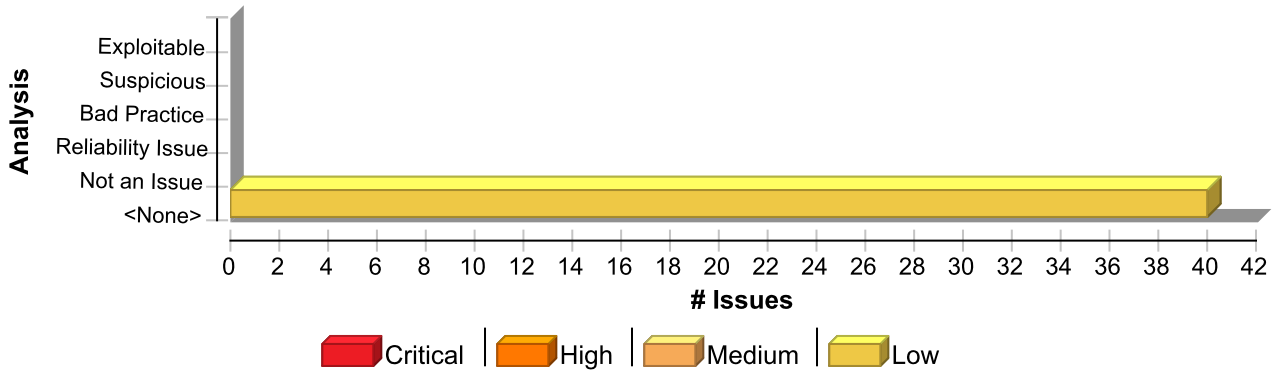
```
public void setUpCalls() {
  boolean firstCall = false;
  boolean secondCall = false;

  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	40	0	0	40
Total	**40**	**0**	**0**	**40**

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorSpec.scala, line 922 (Dead Code: Expression is Always false) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorSpec.scala:922
Taint Flags:

```
919
920 class PersistInRecovery(name: String) extends ExamplePersistentActor(name) {
921 override def receiveRecover = {
922 case Evt("invalid") =>
923 persist(Evt("invalid-recovery"))(updateState)
924 case e: Evt => updateState(e)
925 case RecoveryCompleted =>
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 32 (Dead Code: Expression is Always false) — **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 32 (Dead Code: Expression is Always false) — Low

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:32
Taint Flags:

```
29 }
30
31 val commonBehavior: Receive = {
32 case "boom" => throw new TestException("boom")
33 case GetState => sender() ! events.reverse
34 }
35
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 616 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorSpec.scala:616
Taint Flags:

```
613 var master: ActorRef = _
614
615 val receiveCommand: Receive = {
616 case "Boom" =>
617 master = sender()
618 throw new TestException("boom")
619 }
```

test/scala/akka/persistence/SnapshotSpec.scala, line 63 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotSpec.scala, line 63 (Dead Code: Expression is Always false) — Low

File: test/scala/akka/persistence/SnapshotSpec.scala:63
Taint Flags:

```
60 }
61
62 override def receiveCommand = {
63 case "done" => probe ! "done"
64 case payload: String =>
65 persist(payload) { _ =>
66 probe ! s"${payload}-${lastSequenceNr}"
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 183 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:183
Taint Flags:

```
180 }
181
182 val receiveCommand: Receive = {
183 case Cmd("a") => persist(Evt("a"))(stashWithinHandler)
184 case Cmd("b") => persistAsync(Evt("b"))(stashWithinHandler)
185 case Cmd("c") =>
186 persist(Evt("x")) { _ =>
```

test/scala/akka/persistence/SnapshotSpec.scala, line 43 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/SnapshotSpec.scala:43
Taint Flags:

```
40 }
```

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotSpec.scala, line 43 (Dead Code: Expression is Always false) — Low

```
41
42 override def receiveCommand = {
43 case "done" => probe ! "done"
44 case payload: String =>
45 persist(payload) { _ =>
46 probe ! s"${payload}-${lastSequenceNr}"
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 113 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:113
Taint Flags:

```
110 }
111
112 def unstashBehavior: Receive = {
113 case Cmd("c") =>
114 persist(Evt("c")) { evt =>
115 updateState(evt)
116 context.unbecome()
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 139 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:139
Taint Flags:

```
136
137 val receiveCommand: Receive = commonBehavior.orElse(unstashBehavior).orElse {
138 case Cmd("a") => persistAsync(Evt("a"))(updateState)
139 case Cmd("b") if !stashed =>
```

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 139 (Dead Code: Expression is Always false) — Low

```
140 stash(); stashed = true
141 case Cmd("b") => persistAsync(Evt("b"))(updateState)
142 }
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 141 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:141
Taint Flags:

```
138 case Cmd("a") => persistAsync(Evt("a"))(updateState)
139 case Cmd("b") if !stashed =>
140 stash(); stashed = true
141 case Cmd("b") => persistAsync(Evt("b"))(updateState)
142 }
143
144 override def unstashBehavior: Receive = {
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 151 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:151
Taint Flags:

```
148
149 class AsyncStashingWithinHandlerPersistentActor(name: String) extends AsyncStashingPersistentActor(name) {
150 override def unstashBehavior: Receive = {
151 case Cmd("c") =>
152 persistAsync(Evt("c")) { evt =>
153 updateState(evt); unstashAll()
154 }
```

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 91 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:91
Taint Flags:

```
88
89 class UserStashWithinHandlerManyPersistentActor(name: String) extends UserStashManyPersistentActor(name) {
90 override def unstashBehavior: Receive = {
91 case Cmd("c") =>
92 persist(Evt("c")) { evt =>
93 updateState(evt); context.unbecome(); unstashAll()
94 }
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 45 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:45
Taint Flags:

```
42 var stashed = false
43
44 val receiveCommand: Receive = unstashBehavior.orElse {
45 case Cmd("a") if !stashed =>
46 stash(); stashed = true
47 case Cmd("a") => sender() ! "a"
48 case Cmd("b") => persist(Evt("b"))(evt => sender() ! evt.data)
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 47 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 47 (Dead Code: Expression is Always false) — Low

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:47
Taint Flags:

```
44 val receiveCommand: Receive = unstashBehavior.orElse {
45 case Cmd("a") if !stashed =>
46 stash(); stashed = true
47 case Cmd("a") => sender() ! "a"
48 case Cmd("b") => persist(Evt("b"))(evt => sender() ! evt.data)
49 }
50
```

test/scala/akka/persistence/PerformanceSpec.scala, line 108 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PerformanceSpec.scala:108
Taint Flags:

```
105 })
106
107 val processC: Receive = printProgress.andThen {
108 case "c" =>
109 persist("c")(_ => context.unbecome())
110 unstashAll()
111 case _ => stash()
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 81 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 81 (Dead Code: Expression is Always false) — Low

File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:81
Taint Flags:

```
78 }
79
80 def unstashBehavior: Receive = {
81 case Cmd("c") =>
82 persist(Evt("c")) { evt =>
83 updateState(evt); context.unbecome()
84 }
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 125 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:125
Taint Flags:

```
122 class UserStashWithinHandlerFailureCallbackPersistentActor(name: String)
123 extends UserStashFailurePersistentActor(name) {
124 override def unstashBehavior: Receive = {
125 case Cmd("c") =>
126 persist(Evt("c")) { evt =>
127 updateState(evt)
128 context.unbecome()
```

test/scala/akka/persistence/ManyRecoveriesSpec.scala, line 36 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/ManyRecoveriesSpec.scala:36
Taint Flags:

```
33 persist(Evt(s)) { _ =>
```

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/ManyRecoveriesSpec.scala, line 36 (Dead Code: Expression is Always false) — Low

```
34  sender() ! s"$persistenceId-$s-${lastSequenceNr}"
35  }
36  case "stop" =>
37  context.stop(self)
38  }
39  }
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 821 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorSpec.scala:821
Taint Flags:

```
818  override def persistenceId: String = "StackableTestPersistentActor"
819
820  def receiveCommand = {
821  case "restart" =>
822  throw new Exception("triggering restart") with NoStackTrace {
823  override def toString = "Boom!"
824  }
```

test/scala/akka/persistence/PerformanceSpec.scala, line 103 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PerformanceSpec.scala:103
Taint Flags:

```
100  }
101
102  val receiveCommand: Receive = printProgress.andThen(controlBehavior.orElse {
103  case "a" => persist("a")(_ => context.become(processC))
```

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PerformanceSpec.scala, line 103 (Dead Code: Expression is Always false) — Low

```
104  case "b" => persist("b")(_ => ())
105  })
106
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 48 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:48
Taint Flags:

```
45  case Cmd("a") if !stashed =>
46  stash(); stashed = true
47  case Cmd("a") => sender() ! "a"
48  case Cmd("b") => persist(Evt("b"))(evt => sender() ! evt.data)
49  }
50
51  def unstashBehavior: Receive = {
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 184 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:184
Taint Flags:

```
181
182  val receiveCommand: Receive = {
183  case Cmd("a") => persist(Evt("a"))(stashWithinHandler)
184  case Cmd("b") => persistAsync(Evt("b"))(stashWithinHandler)
185  case Cmd("c") =>
186  persist(Evt("x")) { _ =>
187  }
```

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 67 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:67
Taint Flags:

```
64
65 class UserStashManyPersistentActor(name: String) extends StashExamplePersistentActor(name) {
66 val receiveCommand: Receive = commonBehavior.orElse {
67 case Cmd("a") =>
68 persist(Evt("a")) { evt =>
69 updateState(evt)
70 context.become(processC)
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 436 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorSpec.scala:436
Taint Flags:

```
433
434 class PrimitiveEventPersistentActor(name: String) extends ExamplePersistentActor(name) {
435 val receiveCommand: Receive = {
436 case Cmd("a") => persist(5)(evt => sender() ! evt)
437 }
438 }
439 class PrimitiveEventPersistentActorWithInmemRuntimePluginConfig(name: String, val providedConfig: Config)
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 145 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 145 (Dead Code: Expression is Always false) — Low

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:145
Taint Flags:

```
142 }
143
144 override def unstashBehavior: Receive = {
145 case Cmd("c") => persistAsync(Evt("c"))(updateState); unstashAll()
146 }
147 }
148
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 256 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorSpec.scala:256
Taint Flags:

```
253
254 class ReplyInEventHandlerPersistentActor(name: String) extends ExamplePersistentActor(name) {
255 val receiveCommand: Receive = {
256 case Cmd("a") => persist(Evt("a"))(evt => sender() ! evt.data)
257 }
258 }
259 class ReplyInEventHandlerPersistentActorWithInmemRuntimePluginConfig(name: String, val providedConfig: Config)
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 58 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 58 (Dead Code: Expression is Always false) — Low

File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:58
Taint Flags:

```
55
56 class UserStashWithinHandlerPersistentActor(name: String) extends UserStashPersistentActor(name: String) {
57 override def unstashBehavior: Receive = {
58 case Cmd("c") =>
59 persist(Evt("c")) { evt =>
60 sender() ! evt.data; unstashAll()
61 }
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 73 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:73
Taint Flags:

```
70 context.become(processC)
71 }
72 case Cmd("b-1") => persist(Evt("b-1"))(updateState)
73 case Cmd("b-2") => persist(Evt("b-2"))(updateState)
74 }
75
76 val processC: Receive = unstashBehavior.orElse {
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 42 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorSpec.scala:42
Taint Flags:

```
39 }
```

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorSpec.scala, line 42 (Dead Code: Expression is Always false) — Low

```
40
41 val commonBehavior: Receive = {
42 case "boom" => throw new TestException("boom")
43 case GetState => sender() ! events.reverse
44 case Delete(toSequenceNr) =>
45 persist(Some(sender())) { s =>
```

test/scala/akka/persistence/RecoveryPermitterSpec.scala, line 40 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/RecoveryPermitterSpec.scala:40
Taint Flags:

```
37 throw new TestExc
38 }
39 override def receiveCommand: Receive = {
40 case "stop" =>
41 context.stop(self)
42 }
43 }
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 52 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:52
Taint Flags:

```
49 }
50
51 def unstashBehavior: Receive = {
52 case Cmd("c") => unstashAll(); sender() ! "c"
```

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 52 (Dead Code: Expression is Always false) — Low

```
53 }
54 }
55
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 220 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorSpec.scala:220
Taint Flags:

```
217 def receiveCommand: Receive = commonBehavior.orElse {
218 case c: Cmd => handleCmd(c)
219 case SaveSnapshotSuccess(_) => probe ! "saved"
220 case "snap" => saveSnapshot(events)
221 }
222 }
223 class SnapshottingPersistentActorWithInmemRuntimePluginConfig(
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 72 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:72
Taint Flags:

```
69 updateState(evt)
70 context.become(processC)
71 }
72 case Cmd("b-1") => persist(Evt("b-1"))(updateState)
73 case Cmd("b-2") => persist(Evt("b-2"))(updateState)
74 }
75
```

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 185 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:185
Taint Flags:

```
182 val receiveCommand: Receive = {
183 case Cmd("a") => persist(Evt("a"))(stashWithinHandler)
184 case Cmd("b") => persistAsync(Evt("b"))(stashWithinHandler)
185 case Cmd("c") =>
186 persist(Evt("x")) { _ =>
187 }
188 deferAsync(Evt("c"))(stashWithinHandler)
```

test/scala/akka/persistence/PersistentActorStashingSpec.scala, line 138 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorStashingSpec.scala:138
Taint Flags:

```
135 var stashed = false
136
137 val receiveCommand: Receive = commonBehavior.orElse(unstashBehavior).orElse {
138 case Cmd("a") => persistAsync(Evt("a"))(updateState)
139 case Cmd("b") if !stashed =>
140 stash(); stashed = true
141 case Cmd("b") => persistAsync(Evt("b"))(updateState)
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 244 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/PersistentActorSpec.scala, line 244 (Dead Code: Expression is Always false) — Low

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PersistentActorSpec.scala:244
Taint Flags:

```
241 override def receiveRecover = becomingRecover.orElse(super.receiveRecover)
242
243 val becomingCommand: Receive = receiveCommand.orElse {
244 case "It's changing me" => probe ! "I am becoming"
245 }
246 }
247 class SnapshottingBecomingPersistentActorWithInmemRuntimePluginConfig(
```

test/scala/akka/persistence/PerformanceSpec.scala, line 104 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/PerformanceSpec.scala:104
Taint Flags:

```
101
102 val receiveCommand: Receive = printProgress.andThen(controlBehavior.orElse {
103 case "a" => persist("a")(_ => context.become(processC))
104 case "b" => persist("b")(_ => ())
105 })
106
107 val processC: Receive = printProgress.andThen {
```

test/scala/akka/persistence/SnapshotRecoveryWithEmptyJournalSpec.scala, line 52 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()

Dead Code: Expression is Always false — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotRecoveryWithEmptyJournalSpec.scala, line 52 (Dead Code: Expression is Always false) — Low

File: test/scala/akka/persistence/SnapshotRecoveryWithEmptyJournalSpec.scala:52
Taint Flags:

```
49 }
50
51 override def receiveCommand: PartialFunction[Any, Unit] = {
52 case "done" => probe ! "done"
53 case payload: String =>
54 persist(payload) { _ =>
55 probe ! s"${payload}-${lastSequenceNr}"
```

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSMBase.scala, line 667 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: processEvent()
File: main/scala/akka/persistence/fsm/PersistentFSMBase.scala:667
Taint Flags:

```
664 log.debug("processing {} from {} in state {}", event, srcstr, stateName)
665 }
666
667 if (logDepth > 0) {
668 states(pos) = stateName.asInstanceOf[AnyRef]
669 events(pos) = event
670 advance()
```

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 635 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/fsm/PersistentFSMSpec.scala:635
Taint Flags:

Dead Code: Expression is Always false — Low

Package: akka.persistence.fsm

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 635 (Dead Code: Expression is Always false) — Low

```
632 when(PersistSingleAtOnce) {
633 case Event(i: Int, _) =>
634 stay().applying(IntAdded(i))
635 case Event("4x", _) =>
636 goto(Persist4xAtOnce)
637 case Event(SaveSnapshotSuccess(metadata), _) =>
638 probe ! s"SeqNo=${metadata.sequenceNr}, StateData=${stateData}"
```

test/scala/akka/persistence/fsm/PersistentFSMSpec.scala, line 465 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/fsm/PersistentFSMSpec.scala:465
Taint Flags:

```
462 startWith(LookingAround, EmptyShoppingCart)
463
464 when(LookingAround) {
465 case Event("stay", _) => stay()
466 case Event(_, _) => goto(LookingAround)
467 }
468
```

Dead Code: Expression is Always true (2 issues)

Abstract

This expression will always evaluate to `true`.

Explanation

This expression will always evaluate to `true`; 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 `true`. (The variable `firstCall` is mistakenly used twice.) The result is that the expression `firstCall || secondCall` will always evaluate to `true`, so `setUpForCall()` will always be invoked.

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

  if (fCall < 0) {
    cancelFCall();
    firstCall = false;
  }
  if (sCall < 0) {
    cancelSCall();
    firstCall = false;
  }

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

Example 2: The following method tries to check the variables `firstCall` and `secondCall`. (The variable `firstCall` is mistakenly set to `true` instead of being checked.) The result is that the first part of the expression `firstCall = true && secondCall == true` will always evaluate to `true`.

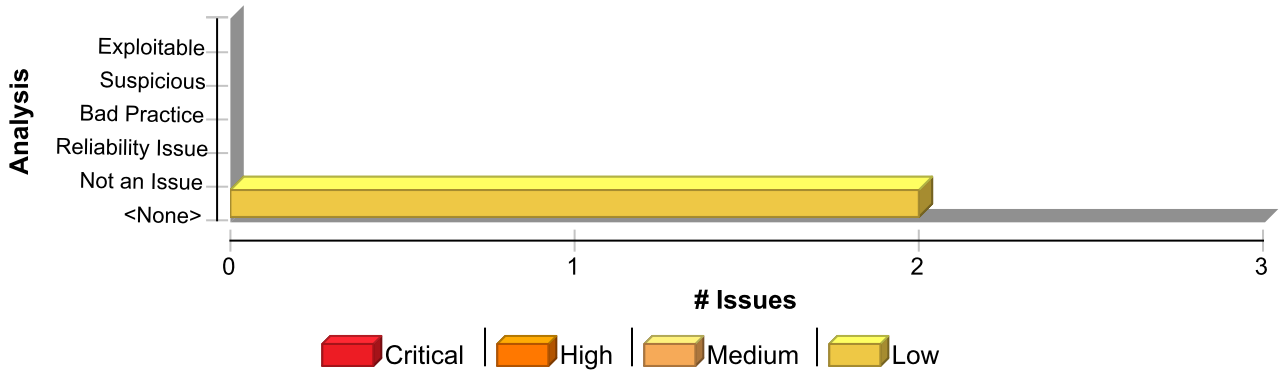
```
public void setUpCalls() {
  boolean firstCall = false;
  boolean secondCall = false;

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

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

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

Dead Code: Expression is Always true — Low

Package: akka.persistence

main/scala/akka/persistence/Eventsourced.scala, line 638 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: main/scala/akka/persistence/Eventsourced.scala:638
Taint Flags:

```
635 }
636
637 {
638 case PersistentRepr(payload, _) if recoveryRunning && _receiveRecover.isDefinedAt(payload) =>
639 _receiveRecover(payload)
640 case s: SnapshotOffer if _receiveRecover.isDefinedAt(s) =>
641 _receiveRecover(s)
```

test/scala/akka/persistence/EventSourcedActorFailureSpec.scala, line 62 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Dead Code: Expression is Always true	Low
Package: akka.persistence	
test/scala/akka/persistence/EventSourcedActorFailureSpec.scala, line 62 (Dead Code: Expression is Always true)	**Low**

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/EventSourcedActorFailureSpec.scala:62
Taint Flags:

```
59 messages.collect {
60 case a: AtomicWrite =>
61 a.payload.collectFirst {
62 case PersistentRepr(Evt(s: String), _: Long) if s.contains("not serializable") => s
63 } match {
64 case Some(s) => Failure(new SimulatedSerializationException(s))
65 case None => AsyncWriteJournal.successUnit
```

Denial of Service (1 issue)

Abstract

An attacker could cause the program to crash or otherwise become unavailable to legitimate users.

Explanation

Attackers may be able to deny service to legitimate users by flooding the application with requests, but flooding attacks can often be defused at the network layer. More problematic are bugs that allow an attacker to overload the application using a small number of requests. Such bugs allow the attacker to specify the quantity of system resources their requests will consume or the duration for which they will use them. **Example 1:** The following code allows a user to specify the amount of time for which a thread will sleep. By specifying a large number, an attacker may tie up the thread indefinitely. With a small number of requests, the attacker may deplete the application's thread pool.

```
  int usrSleepTime = Integer.parseInt(usrInput);
  Thread.sleep(usrSleepTime);
```

Example 2: The following code reads a String from a zip file. Because it uses the `readLine()` method, it will read an unbounded amount of input. An attacker may take advantage of this code to cause an `OutOfMemoryException` or to consume a large amount of memory so that the program spends more time performing garbage collection or runs out of memory during some subsequent operation.

```
  InputStream zipInput = zipFile.getInputStream(zipEntry);
  Reader zipReader = new InputStreamReader(zipInput);
  BufferedReader br = new BufferedReader(zipReader);
  String line = br.readLine();
```

Recommendation

Validate user input to ensure that it will not cause inappropriate resource utilization. **Example 3:** The following code allows a user to specify the amount of time for which a thread will sleep just as in `Example 1`, but only if the value is within reasonable bounds.

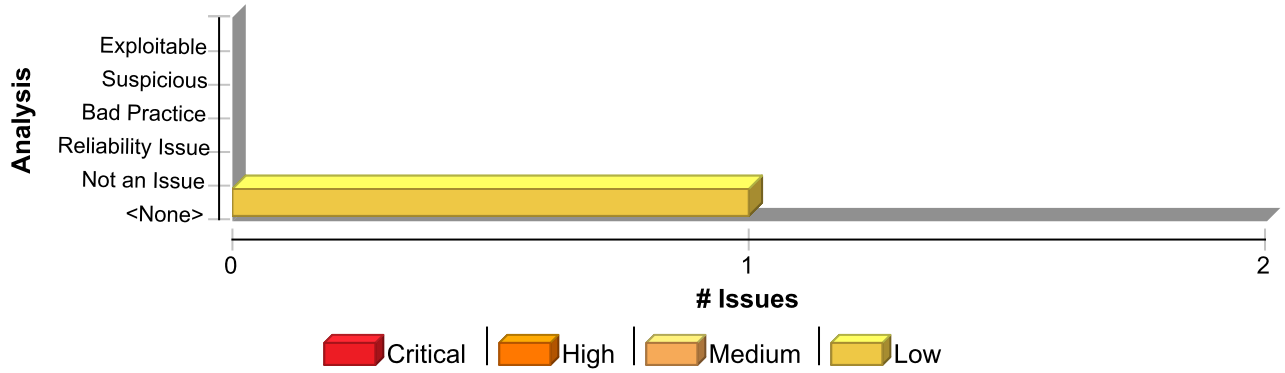
```
  int usrSleepTime = Integer.parseInt(usrInput);
  if (usrSleepTime >= SLEEP_MIN &&
      usrSleepTime <= SLEEP_MAX) {
    Thread.sleep(usrSleepTime);
  } else {
    throw new Exception("Invalid sleep duration");
  }
}
```

Example 4: The following code reads a String from a zip file just as in `Example 2`, but the maximum string length it will read is `MAX_STR_LEN` characters.

```
  InputStream zipInput = zipFile.getInputStream(zipEntry);
  Reader zipReader = new InputStreamReader(zipInput);
  BufferedReader br = new BufferedReader(zipReader);
  StringBuffer sb = new StringBuffer();
  int intC;
  while ((intC = br.read()) != -1) {
    char c = (char) intC;
    if (c == '\n') {
      break;
    }
    if (sb.length() >= MAX_STR_LEN) {
      throw new Exception("input too long");
    }
    sb.append(c);
  }
```

```
String line = sb.toString();
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Denial of Service	1	0	0	1
Total	**1**	**0**	**0**	**1**

Denial of Service — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotSerializationSpec.scala, line 47 (Denial of Service) — Low

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: readLine()
Enclosing Method: fromBinary()
File: test/scala/akka/persistence/SnapshotSerializationSpec.scala:47
Taint Flags:

```
44 def fromBinary(bytes: Array[Byte], clazz: Option[Class[_]]): AnyRef = {
45 val bStream = new ByteArrayInputStream(bytes)
46 val reader = new BufferedReader(new InputStreamReader(bStream))
47 new MySnapshot(reader.readLine())
48 }
49 }
50
```

Insecure Randomness (5 issues)

Abstract

Standard pseudorandom number generators cannot withstand cryptographic attacks.

Explanation

Insecure randomness errors occur when a function that can produce predictable values is used as a source of randomness in a security-sensitive context. Computers are deterministic machines, and as such are unable to produce true randomness. Pseudorandom Number Generators (PRNGs) approximate randomness algorithmically, starting with a seed from which subsequent values are calculated. There are two types of PRNGs: statistical and cryptographic. Statistical PRNGs provide useful statistical properties, but their output is highly predictable and form an easy to reproduce numeric stream that is unsuitable for use in cases where security depends on generated values being unpredictable. Cryptographic PRNGs address this problem by generating output that is more difficult to predict. For a value to be cryptographically secure, it must be impossible or highly improbable for an attacker to distinguish between the generated random value and a truly random value. In general, if a PRNG algorithm is not advertised as being cryptographically secure, then it is probably a statistical PRNG and should not be used in security-sensitive contexts, where its use can lead to serious vulnerabilities such as easy-to-guess temporary passwords, predictable cryptographic keys, session hijacking, and DNS spoofing. **Example:** The following code uses a statistical PRNG to create a URL for a receipt that remains active for some period of time after a purchase.

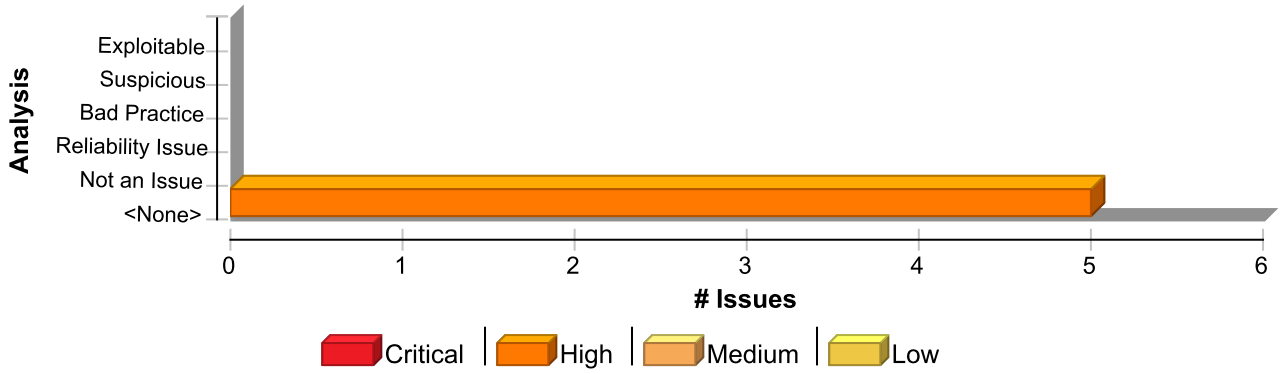
```
String GenerateReceiptURL(String baseUrl) {
    Random ranGen = new Random();
    ranGen.setSeed((new Date()).getTime());
    return (baseUrl + ranGen.nextInt(400000000) + ".html");
}
```

This code uses the `Random.nextInt()` function to generate "unique" identifiers for the receipt pages it generates. Since `Random.nextInt()` is a statistical PRNG, it is easy for an attacker to guess the strings it generates. Although the underlying design of the receipt system is also faulty, it would be more secure if it used a random number generator that did not produce predictable receipt identifiers, such as a cryptographic PRNG.

Recommendation

When unpredictability is critical, as is the case with most security-sensitive uses of randomness, use a cryptographic PRNG. Regardless of the PRNG you choose, always use a value with sufficient entropy to seed the algorithm. (Do not use values such as the current time because it offers only negligible entropy.) The Java language provides a cryptographic PRNG in `java.security.SecureRandom`. As is the case with other algorithm-based classes in `java.security`, `SecureRandom` provides an implementation-independent wrapper around a particular set of algorithms. When you request an instance of a `SecureRandom` object using `SecureRandom.getInstance()`, you can request a specific implementation of the algorithm. If the algorithm is available, then it is given as a `SecureRandom` object. If it is unavailable or if you do not specify a particular implementation, then you are given a `SecureRandom` implementation selected by the system. Sun provides a single `SecureRandom` implementation with the Java distribution named `SHA1PRNG`, which Sun describes as computing: "The SHA-1 hash over a true-random seed value concatenated with a 64-bit counter which is incremented by 1 for each operation. From the 160-bit SHA-1 output, only 64 bits are used [1]." However, the specifics of the Sun implementation of the `SHA1PRNG` algorithm are poorly documented, and it is unclear what sources of entropy the implementation uses and therefore what amount of true randomness exists in its output. Although there is speculation on the Web about the Sun implementation, there is no evidence to contradict the claim that the algorithm is cryptographically strong and can be used safely in security-sensitive contexts.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Insecure Randomness	5	0	0	5
Total	**5**	**0**	**0**	**5**

Insecure Randomness — High

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala, line 63 (Insecure Randomness) — **High**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextDouble()
Enclosing Method: shouldFail()
File: test/scala/akka/persistence/AtLeastOnceDeliveryFailureSpec.scala:63
Taint Flags:

```
60  }
61
62  def shouldFail(rate: Double) =
63  random.nextDouble() < rate
64  }
65
66  class ChaosSender(destination: ActorRef, val probe: ActorRef)
```

Package: akka.persistence.journal.chaos

test/scala/akka/persistence/journal/chaos/ChaosJournal.scala, line 69 (Insecure Randomness) — **High**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Insecure Randomness — High

Package: akka.persistence.journal.chaos

test/scala/akka/persistence/journal/chaos/ChaosJournal.scala, line 69 (Insecure Randomness) — High

Sink: nextInt()
Enclosing Method: asyncReplayMessages()
File: test/scala/akka/persistence/journal/chaos/ChaosJournal.scala:69
Taint Flags:

```
66 replayCallback: (PersistentRepr) => Unit): Future[Unit] =
67 if (shouldFail(replayFailureRate)) {
68 val rm = read(persistenceId, fromSequenceNr, toSequenceNr, max)
69 val sm = rm.take(random.nextInt(rm.length + 1)).map(_._1)
70 sm.foreach(replayCallback)
71 Future.failed(new ReplayFailedException(sm))
72 } else {
```

test/scala/akka/persistence/journal/chaos/ChaosJournal.scala, line 82 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextDouble()
Enclosing Method: shouldFail()
File: test/scala/akka/persistence/journal/chaos/ChaosJournal.scala:82
Taint Flags:

```
79 else Future.successful(highestSequenceNr(persistenceId))
80
81 def shouldFail(rate: Double): Boolean =
82 random.nextDouble() < rate
83 }
84
85 undefined
```

Package: akka.persistence.journal.leveldb

test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala, line 208 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextString()
Enclosing Method: akka$persistence$journal$leveldb$JournalCompactionSpec$EventLogger$$randomText()

Insecure Randomness	High

Package: akka.persistence.journal.leveldb

test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala, line 208 (Insecure Randomness)	High

File: test/scala/akka/persistence/journal/leveldb/JournalCompactionSpec.scala:208
Taint Flags:

```
205 watcher ! Generated(evt.seqNr)
206 }
207
208 private def randomText(): String = Random.nextString(1024)
209 }
210
211 }
```

Package: test.scala.akka.persistence

test/scala/akka/persistence/PersistentActorSpec.scala, line 1234 (Insecure Randomness)	High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: apply()
File: test/scala/akka/persistence/PersistentActorSpec.scala:1234
Taint Flags:

```
1231 }
1232
1233 commands.foreach { i =>
1234 Thread.sleep(Random.nextInt(10))
1235 persistentActor ! i
1236 }
1237
```

J2EE Bad Practices: Threads (3 issues)

Abstract

Thread management in a web application is forbidden in some circumstances and is always highly error prone.

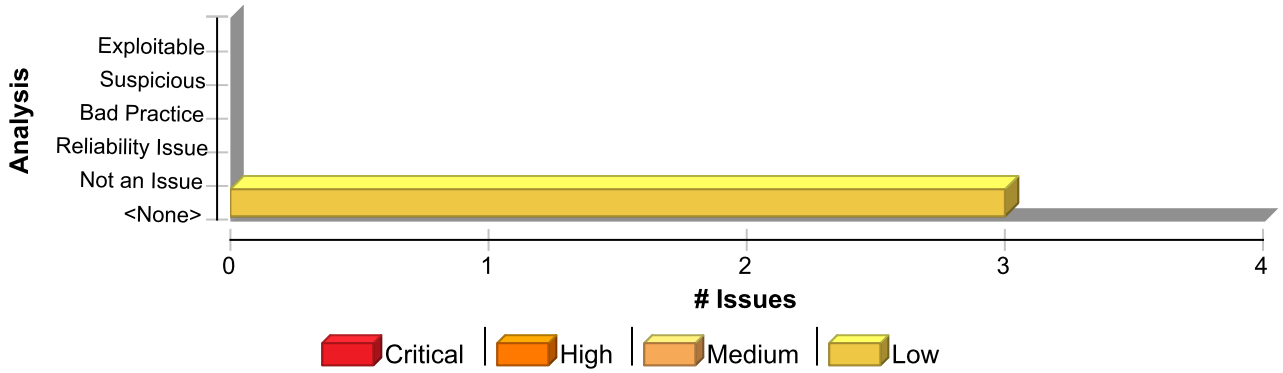
Explanation

Thread management in a web application is forbidden by the J2EE standard in some circumstances and is always highly error prone. Managing threads is difficult and is likely to interfere in unpredictable ways with the behavior of the application container. Even without interfering with the container, thread management usually leads to bugs that are hard to detect and diagnose like deadlock, race conditions, and other synchronization errors.

Recommendation

Avoid managing threads directly from within the web application. Instead use standards such as message driven beans and the EJB timer service that are provided by the application container.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: Threads	3	0	0	3
Total	**3**	**0**	**0**	**3**

J2EE Bad Practices: Threads	Low
Package: test.scala.akka.persistence	
test/scala/akka/persistence/PersistentActorSpec.scala, line 325 (J2EE Bad Practices: Threads)	**Low**

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: sleep()
Enclosing Method: apply()
File: test/scala/akka/persistence/PersistentActorSpec.scala:325

J2EE Bad Practices: Threads — Low

Package: test.scala.akka.persistence

test/scala/akka/persistence/PersistentActorSpec.scala, line 325 (J2EE Bad Practices: Threads) — Low

Taint Flags:

```
322
323 persistAsync(event) { evt =>
324 // be way slower, in order to be overtaken by the other callback
325 Thread.sleep(300)
326 sender() ! s"${evt.data}-a-${sendMsgCounter.incrementAndGet()}"
327 }
328 persistAsync(event) { evt =>
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 771 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: sleep()
Enclosing Method: apply()
File: test/scala/akka/persistence/PersistentActorSpec.scala:771
Taint Flags:

```
768 probe ! outer
769 persist(s + "-inner") { inner =>
770 probe ! inner
771 Thread.sleep(1000) // really long wait here...
772 // the next incoming command must be handled by the following function
773 context.become({ case _ => sender() ! "done" })
774 }
```

test/scala/akka/persistence/PersistentActorSpec.scala, line 1234 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: sleep()
Enclosing Method: apply()
File: test/scala/akka/persistence/PersistentActorSpec.scala:1234
Taint Flags:

```
1231 }
1232
```

J2EE Bad Practices: Threads	Low

Package: test.scala.akka.persistence

test/scala/akka/persistence/PersistentActorSpec.scala, line 1234 (J2EE Bad Practices: Threads)	Low

```
1233 commands.foreach { i =>
1234 Thread.sleep(Random.nextInt(10))
1235 persistentActor ! i
1236 }
1237
```

Key Management: Hardcoded Encryption Key (4 issues)

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:

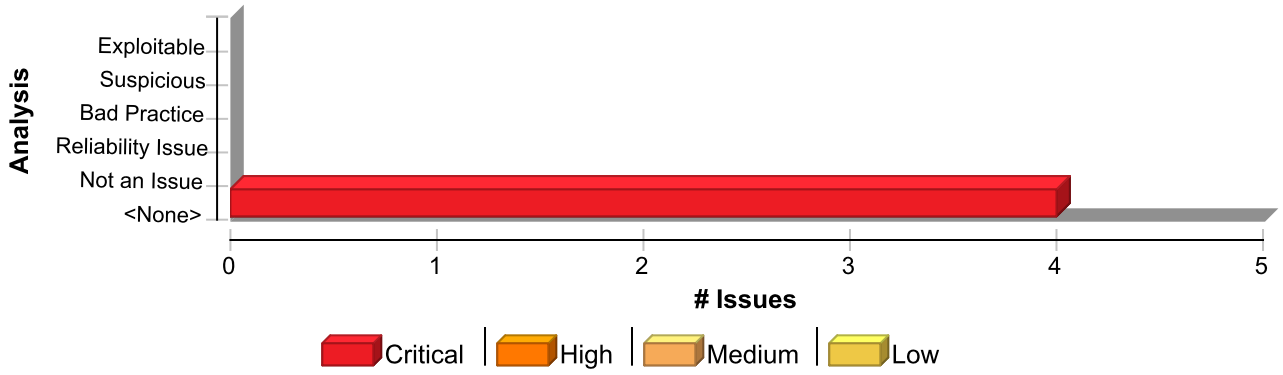
```
...
private static final String encryptionKey = "lakdsljkalkjlksdfkl";
byte[] keyBytes = encryptionKey.getBytes();
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher encryptCipher = Cipher.getInstance("AES");
encryptCipher.init(Cipher.ENCRYPT_MODE, key);
...
```

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	4	0	0	4
Total	**4**	**0**	**0**	**4**

Key Management: Hardcoded Encryption Key — Critical

Package: akka.persistence.fsm

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 41 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FieldAccess: key
Enclosing Method: SnapshotAfter()
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:41
Taint Flags:

```
38  * See `akka.persistence.fsm.snapshot-after` for configuration options.
39  */
40 private[akka] class SnapshotAfter(config: Config) extends Extension {
41  val key = "akka.persistence.fsm.snapshot-after"
42  val snapshotAfterValue = config.getString(key).toLowerCase match {
43  case "off" => None
44  case _ => Some(config.getInt(key))
```

main/scala/akka/persistence/fsm/PersistentFSM.scala, line 41 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: FieldAccess: key
Enclosing Method: SnapshotAfter()
File: main/scala/akka/persistence/fsm/PersistentFSM.scala:41
Taint Flags:

```
38  * See `akka.persistence.fsm.snapshot-after` for configuration options.
39  */
40 private[akka] class SnapshotAfter(config: Config) extends Extension {
41  val key = "akka.persistence.fsm.snapshot-after"
42  val snapshotAfterValue = config.getString(key).toLowerCase match {
43  case "off" => None
44  case _ => Some(config.getInt(key))
```

Package: akka.persistence.journal.inmem

main/scala/akka/persistence/journal/inmem/InmemJournal.scala, line 64 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features

Key Management: Hardcoded Encryption Key — Critical

Package: akka.persistence.journal.inmem

main/scala/akka/persistence/journal/inmem/InmemJournal.scala, line 64 (Key Management: Hardcoded Encryption Key) — Critical

Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: key
Enclosing Method: InmemJournal()
File: main/scala/akka/persistence/journal/inmem/InmemJournal.scala:64
Taint Flags:

```
61 private val log = Logging(context.system, classOf[InmemJournal])
62
63 private val testSerialization = {
64 val key = "test-serialization"
65 if (cfg.hasPath(key)) cfg.getBoolean("test-serialization")
66 else false
67 }
```

main/scala/akka/persistence/journal/inmem/InmemJournal.scala, line 64 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: key
Enclosing Method: InmemJournal()
File: main/scala/akka/persistence/journal/inmem/InmemJournal.scala:64
Taint Flags:

```
61 private val log = Logging(context.system, classOf[InmemJournal])
62
63 private val testSerialization = {
64 val key = "test-serialization"
65 if (cfg.hasPath(key)) cfg.getBoolean("test-serialization")
66 else false
67 }
```

Object Model Violation: Just one of equals() and hashCode() Defined (1 issue)

Abstract

This class overrides only one of `equals()` and `hashCode()`.

Explanation

Java objects are expected to obey a number of invariants related to equality. One of these invariants is that equal objects must have equal hashcodes. In other words, if `a.equals(b) == true` then `a.hashCode() == b.hashCode()`. Failure to uphold this invariant is likely to cause trouble if objects of this class are stored in a collection. If the objects of the class in question are used as a key in a Hashtable or if they are inserted into a Map or Set, it is critical that equal objects have equal hashcodes. **Example 1:** The following class overrides `equals()` but not `hashCode()`.

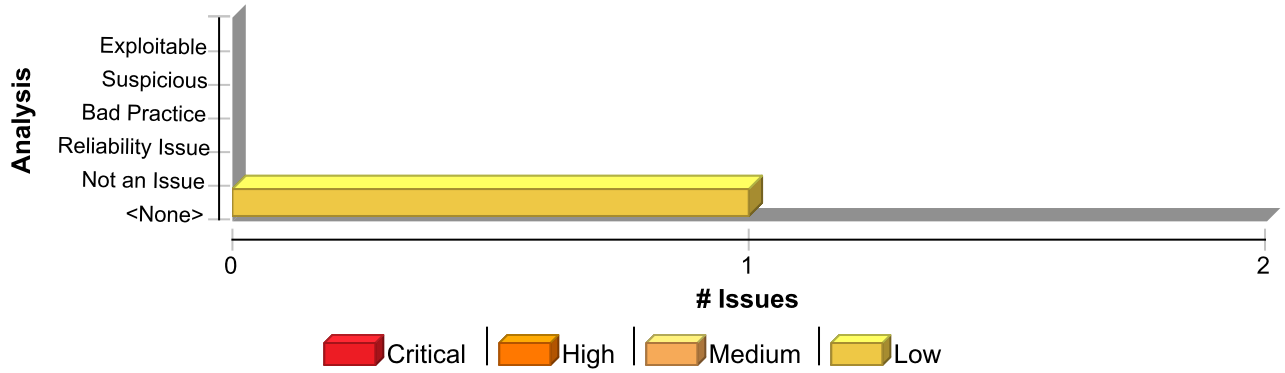
```
public class halfway() {
  public boolean equals(Object obj) {
    ...
  }
}
```

Recommendation

The FindBugs documentation recommends the following simple "starter" implementation of `hashCode()` [1]. It is highly inefficient, but it will produce correct results. If you do not believe that `hashCode()` is important for your program, consider using this implementation. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
public class halfway() {
  public boolean equals(Object obj) {
    ...
  }

public int hashCode() {
    assert false : "hashCode not designed";
    return 42; // any arbitrary constant will do
  }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Object Model Violation: Just one of equals() and hashCode() Defined	1	0	0	1
Total	**1**	**0**	**0**	**1**

Object Model Violation: Just one of equals() and hashCode() Defined — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotSerializationSpec.scala, line 20 (Object Model Violation: Just one of equals() and hashCode() Defined) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: equals
Enclosing Method: equals()
File: test/scala/akka/persistence/SnapshotSerializationSpec.scala:20
Taint Flags:

```
17 // is bigger than 255 bytes (this happens to be 269)
18 object XXXXXXXXXXXXXXXXXXX {
19 class MySnapshot(val id: String) extends SerializationMarker {
20 override def equals(obj: scala.Any) = obj match {
21 case s: MySnapshot => s.id.equals(id)
22 case _ => false
23 }
```

Poor Error Handling: Overly Broad Catch (1 issue)

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:

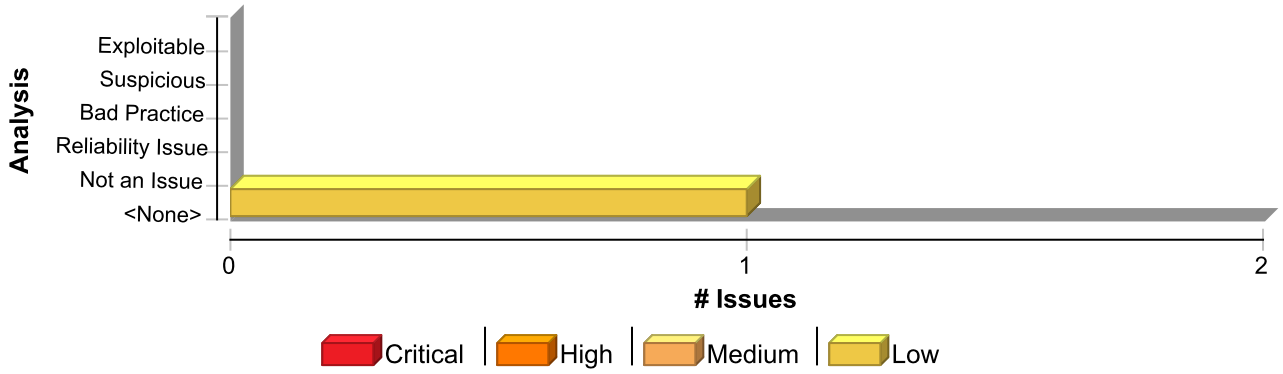
```
  try {
    doExchange();
  }
  catch (Exception e) {
    logger.error("doExchange failed", e);
  }
```

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	1	0	0	1
Total	**1**	**0**	**0**	**1**

Poor Error Handling: Overly Broad Catch — Low

Package: akka.persistence

test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala, line 182 (Poor Error Handling: Overly Broad Catch) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: applyOrElse()
File: test/scala/akka/persistence/AtLeastOnceDeliverySpec.scala:182
Taint Flags:

```
179 case any =>
180 // this is not supported currently, so expecting exception
181 try deliver(context.actorSelection("*"))(id => s"$any$id")
182 catch { case ex: Exception => sender() ! Failure(ex) }
183 }
184
185 override def receiveRecover = Actor.emptyBehavior
```

System Information Leak (1 issue)

Abstract

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

Explanation

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

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

Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a remote user. For example, with scripting mechanisms it is trivial to redirect output information from "Standard error" or "Standard output" into a file or another program. Alternatively, the system that the program runs on could have a remote logging mechanism such as a "syslog" server that sends the logs to a remote device. During development, you have no way of knowing where this information might end up being displayed. In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In `Example 1`, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program. Information leaks are also a concern in a mobile computing environment. With mobile platforms, applications are downloaded from various sources and are run alongside each other on the same device. The likelihood of running a piece of malware next to a banking application is high, which is why application authors need to be careful about what information they include in messages addressed to other applications running on the device. **Example 2:** The following code broadcasts the stack trace of a caught exception to all the registered Android receivers.

```
...
try {
  ...
} catch (Exception e) {
    String exception = Log.getStackTraceString(e);
    Intent i = new Intent();
    i.setAction("SEND_EXCEPTION");
    i.putExtra("exception", exception);
    view.getContext().sendBroadcast(i);
}
...
```

This is another scenario specific to the mobile environment. Most mobile devices now implement a Near-Field Communication (NFC) protocol for quickly sharing information between devices using radio communication. It works by bringing devices in close proximity or having the devices touch each other. Even though the communication range of NFC is limited to just a few centimeters, eavesdropping, data modification and various other types of attacks are possible, because NFC alone does not ensure secure communication. **Example 3:** The Android platform provides support for NFC. The following code creates a message that gets pushed to the other device within range.

```
...
public static final String TAG = "NfcActivity";
private static final String DATA_SPLITTER = "__:DATA:__";
private static final String MIME_TYPE = "application/my.applications.mimetype";
...
TelephonyManager tm =
(TelephonyManager)Context.getSystemService(Context.TELEPHONY_SERVICE);
String VERSION = tm.getDeviceSoftwareVersion();
```

```
...
NfcAdapter nfcAdapter = NfcAdapter.getDefaultAdapter(this);
if (nfcAdapter == null)
  return;

String text = TAG + DATA_SPLITTER + VERSION;
NdefRecord record = new NdefRecord(NdefRecord.TNF_MIME_MEDIA,
            MIME_TYPE.getBytes(), new byte[0], text.getBytes());
NdefRecord[] records = { record };
NdefMessage msg = new NdefMessage(records);
nfcAdapter.setNdefPushMessage(msg, this);
...
```

An NFC Data Exchange Format (NDEF) message contains typed data, a URI, or a custom application payload. If the message contains information about the application, such as its name, MIME type, or device software version, this information could be leaked to an eavesdropper.

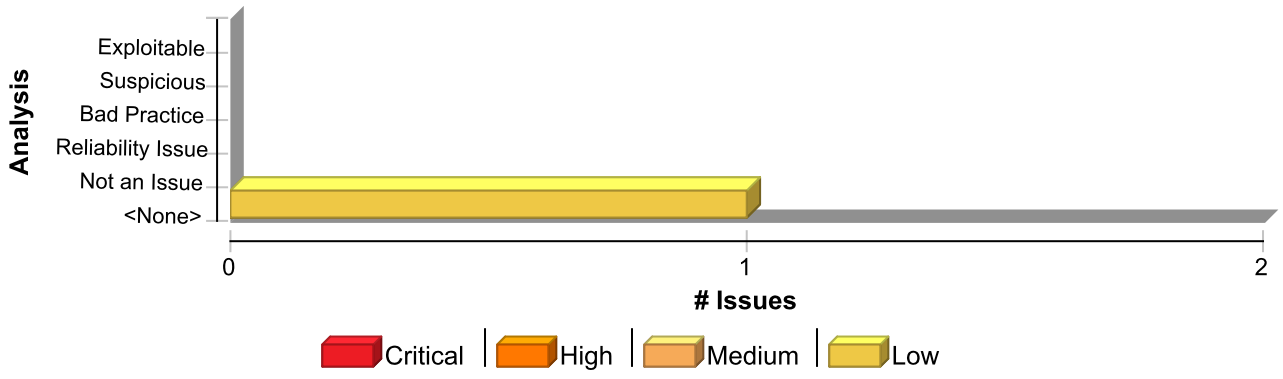
Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. Debug traces can sometimes appear in non-obvious places (embedded in comments in the HTML for an error page, for example). Even brief error messages that do not reveal stack traces or database dumps can potentially aid an attacker. For example, an "Access Denied" message can reveal that a file or user exists on the system. Because of this, never send information to a resource directly outside the program. **Example 4:** The following code broadcasts the stack trace of a caught exception within your application only, so that it cannot be leaked to other apps on the system. Additionally, this technique is more efficient than globally broadcasting through the system.

```
...
try {
  ...
} catch (Exception e) {
    String exception = Log.getStackTraceString(e);
    Intent i = new Intent();
    i.setAction("SEND_EXCEPTION");
    i.putExtra("exception", exception);
    LocalBroadcastManager.getInstance(view.getContext()).sendBroadcast(i);
}
...
```

If you are concerned about leaking system data via NFC on an Android device, you could do one of the following three things. Do not include system data in the messages pushed to other devices in range, encrypt the payload of the message, or establish a secure communication channel at a higher layer.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak	1	0	0	1
Total	**1**	**0**	**0**	**1**

System Information Leak	Low
Package: main.scala.akka.persistence.snapshot.local	
main/scala/akka/persistence/snapshot/local/LocalSnapshotStore.scala, line 164 (System Information Leak)	**Low**

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.io.File.listFiles()
From: akka.persistence.snapshot.local.LocalSnapshotStore.snapshotMetadatas
File: main/scala/akka/persistence/snapshot/local/LocalSnapshotStore.scala:158

```
155  private def snapshotMetadatas(
156  persistenceId: String,
157  criteria: SnapshotSelectionCriteria): immutable.Seq[SnapshotMetadata] = {
158  val files = snapshotDir().listFiles(new SnapshotFilenameFilter(persistenceId))
159  if (files eq null) Nil // if the dir was removed
160  else {
161  files
```

Sink Details

Sink: scala.MatchError.MatchError()
Enclosing Method: apply()
File: main/scala/akka/persistence/snapshot/local/LocalSnapshotStore.scala:164
Taint Flags: NUMBER, PRIMARY_KEY, SYSTEMINFO

```
161  files
162  .map(_.getName)
163  .flatMap { filename =>
164  extractMetadata(filename).map {
165  case (pid, snr, tms) => SnapshotMetadata(URLDecoder.decode(pid, UTF_8), snr, tms)
166  }
167  }
```

Unchecked Return Value (3 issues)

Abstract

Ignoring a method's return value can cause the program to overlook unexpected states and conditions.

Explanation

It is not uncommon for Java programmers to misunderstand `read()` and related methods that are part of many `java.io` classes. Most errors and unusual events in Java result in an exception being thrown. (This is one of the advantages that Java has over languages like C: Exceptions make it easier for programmers to think about what can go wrong.) But the stream and reader classes do not consider it unusual or exceptional if only a small amount of data becomes available. These classes simply add the small amount of data to the return buffer, and set the return value to the number of bytes or characters read. There is no guarantee that the amount of data returned is equal to the amount of data requested. This behavior makes it important for programmers to examine the return value from `read()` and other IO methods to ensure that they receive the amount of data they expect. **Example:** The following code loops through a set of users, reading a private data file for each user. The programmer assumes that the files are always exactly 1 kilobyte in size and therefore ignores the return value from `read()`. If an attacker can create a smaller file, the program will recycle the remainder of the data from the previous user and handle it as though it belongs to the attacker.

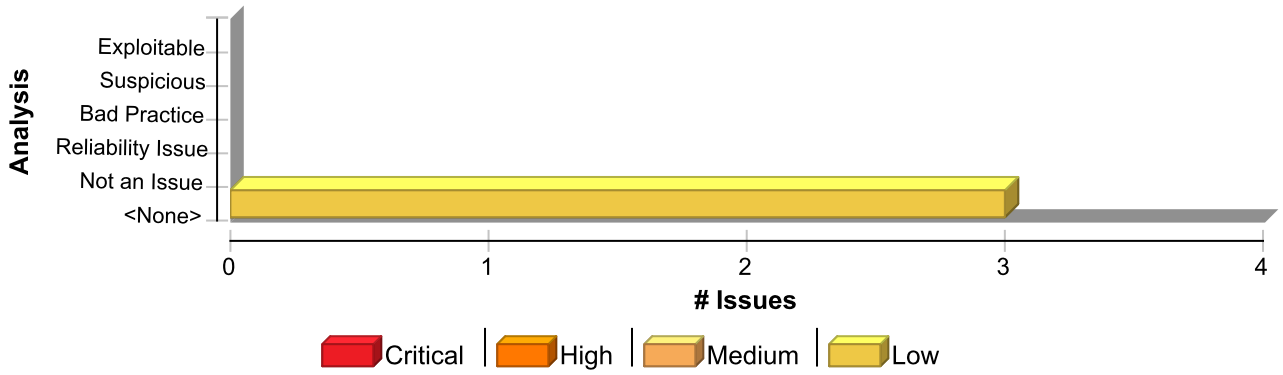
```
FileInputStream fis;
byte[] byteArray = new byte[1024];
for (Iterator i=users.iterator(); i.hasNext();) {
    String userName = (String) i.next();
    String pFileName = PFILE_ROOT + "/" + userName;
    FileInputStream fis = new FileInputStream(pFileName);
    fis.read(byteArray); // the file is always 1k bytes
    fis.close();
    processPFile(userName, byteArray);
}
```

Recommendation

```
  FileInputStream fis;
  byte[] byteArray = new byte[1024];
  for (Iterator i=users.iterator(); i.hasNext();) {
    String userName = (String) i.next();
    String pFileName = PFILE_ROOT + "/" + userName;
    fis = new FileInputStream(pFileName);
    int bRead = 0;
    while (bRead < 1024) {
        int rd = fis.read(byteArray, bRead, 1024 - bRead);
        if (rd == -1) {
          throw new IOException("file is unusually small");
        }
        bRead += rd;
    }
    // could add check to see if file is too large here
    fis.close();
    processPFile(userName, byteArray);
  }
```

Note: Because the fix for this problem is relatively complicated, you might be tempted to use a simpler approach, such as checking the size of the file before you begin reading. Such an approach would render the application vulnerable to a file system race condition, whereby an attacker could replace a well-formed file with a malicious file between the file size check and the call to read data from the file.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unchecked Return Value	3	0	0	3
Total	**3**	**0**	**0**	**3**

Unchecked Return Value — Low

Package: akka.persistence

test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala, line 82 (Unchecked Return Value) — **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: renameTo()
Enclosing Method: save()
File: test/scala/akka/persistence/SnapshotFailureRobustnessSpec.scala:82
Taint Flags:

```
79 if (metadata.sequenceNr == 2 || snapshot.toString.startsWith("boom")) {
80 val bytes = "b0rkb0rk".getBytes("UTF-8") // length >= 8 to prevent EOF exception
81 val tmpFile = withOutputStream(metadata)(_.write(bytes))
82 tmpFile.renameTo(snapshotFileForWrite(metadata))
83 } else super.save(metadata, snapshot)
84 }
85 }
```

Package: akka.persistence.serialization

main/scala/akka/persistence/serialization/SnapshotSerializer.scala, line 69 (Unchecked Return Value) — **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Unchecked Return Value	Low

Package: akka.persistence.serialization

main/scala/akka/persistence/serialization/SnapshotSerializer.scala, line 69 (Unchecked Return Value)	Low

Sink Details

Sink: read()
Enclosing Method: headerFromBinary()
File: main/scala/akka/persistence/serialization/SnapshotSerializer.scala:69
Taint Flags:

```
66 if (remaining == 0) ""
67 else {
68 val manifestBytes = new Array[Byte](remaining)
69 in.read(manifestBytes)
70 new String(manifestBytes, UTF_8)
71 }
72 (serializerId, manifest)
```

Package: akka.persistence.snapshot.local

main/scala/akka/persistence/snapshot/local/LocalSnapshotStore.scala, line 123 (Unchecked Return Value)	Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: renameTo()
Enclosing Method: save()
File: main/scala/akka/persistence/snapshot/local/LocalSnapshotStore.scala:123
Taint Flags:

```
120
121 protected def save(metadata: SnapshotMetadata, snapshot: Any): Unit = {
122 val tmpFile = withOutputStream(metadata)(serialize(_, Snapshot(snapshot)))
123 tmpFile.renameTo(snapshotFileForWrite(metadata))
124 }
125
126 protected def deserialize(inputStream: InputStream): Snapshot =
```