Fortify Standalone Report Generator

Developer Workbook

akka-actor

Table of Contents

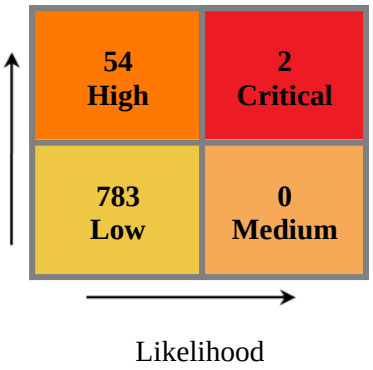
Executive Summary

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

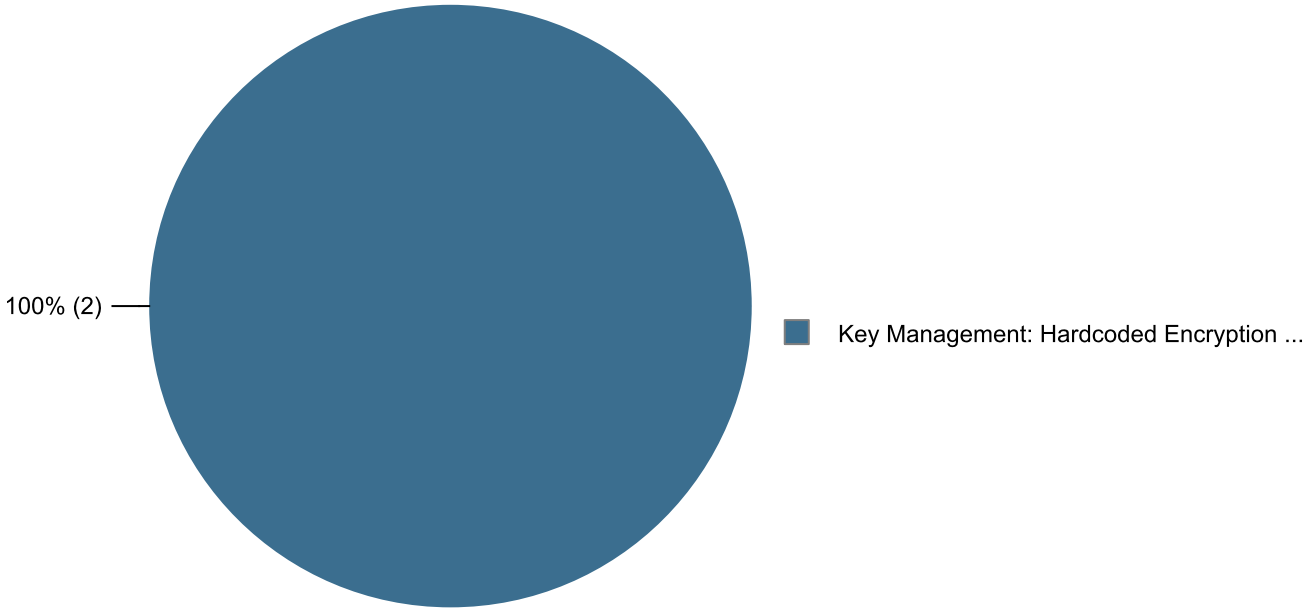
This section provides an overview of the issues uncovered during analysis.

Project Name:	akka-actor
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, 10:58 AM	**Engine Version:**	21.1.1.0009
Host Name:	Jacks-Work-MBP.local	**Certification:**	VALID
Number of Files:	201	**Lines of Code:**	16,363

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	
Access Specifier Manipulation	0	0 / 7	0	0	0 / 7
Code Correctness: Class Does Not Implement Cloneable	0	0	0	0 / 3	0 / 3
Code Correctness: Class Does Not Implement equals	0	0	0	0 / 135	0 / 135
Code Correctness: Constructor Invokes Overridable Function	0	0	0	0 / 316	0 / 316
Code Correctness: Double-Checked Locking	0	0 / 1	0	0	0 / 1
Code Correctness: Erroneous String Compare	0	0	0	0 / 58	0 / 58
Code Correctness: Incorrect Serializable Method Signature	0	0	0	0 / 2	0 / 2
Code Correctness: Non-Static Inner Class Implements Serializable	0	0	0	0 / 77	0 / 77
Code Correctness: Non-Synchronized Method Overrides Synchronized Method	0	0	0	0 / 5	0 / 5
Code Correctness: readObject() Invokes Overridable Function	0	0	0	0 / 1	0 / 1
Dead Code: Expression is Always false	0	0	0	0 / 20	0 / 20
Dead Code: Expression is Always true	0	0	0	0 / 9	0 / 9
Denial of Service	0	0	0	0 / 1	0 / 1
Insecure Randomness	0	0 / 9	0	0	0 / 9
J2EE Bad Practices: JVM Termination	0	0	0	0 / 4	0 / 4
J2EE Bad Practices: Leftover Debug Code	0	0	0	0 / 1	0 / 1
J2EE Bad Practices: Sockets	0	0	0	0 / 8	0 / 8
J2EE Bad Practices: Threads	0	0	0	0 / 58	0 / 58
Key Management: Hardcoded Encryption Key	0 / 2	0	0	0	0 / 2
Missing Check against Null	0	0	0	0 / 4	0 / 4
Null Dereference	0	0 / 3	0	0	0 / 3
Object Model Violation: Erroneous clone() Method	0	0	0	0 / 3	0 / 3
Object Model Violation: Just one of equals() and hashCode() Defined	0	0	0	0 / 1	0 / 1
Often Misused: Authentication	0	0 / 9	0	0	0 / 9
Poor Error Handling: Empty Catch Block	0	0	0	0 / 4	0 / 4
Poor Error Handling: Throw Inside Finally	0	0	0	0 / 1	0 / 1
Poor Logging Practice: Use of a System Output Stream	0	0	0	0 / 12	0 / 12
Poor Style: Confusing Naming	0	0	0	0 / 2	0 / 2
Poor Style: Value Never Read	0	0	0	0 / 10	0 / 10
Redundant Null Check	0	0	0	0 / 3	0 / 3
System Information Leak	0	0	0	0 / 5	0 / 5
System Information Leak: External	0	0	0	0 / 33	0 / 33
System Information Leak: Internal	0	0	0	0 / 5	0 / 5
Unchecked Return Value	0	0	0	0 / 1	0 / 1
Unreleased Resource: Streams	0	0 / 2	0	0	0 / 2
Unreleased Resource: Synchronization	0	0 / 23	0	0	0 / 23
Unsafe Reflection	0	0	0	0 / 1	0 / 1

Results Outline

Access Specifier Manipulation (7 issues)

Abstract

The method call changes an access specifier.

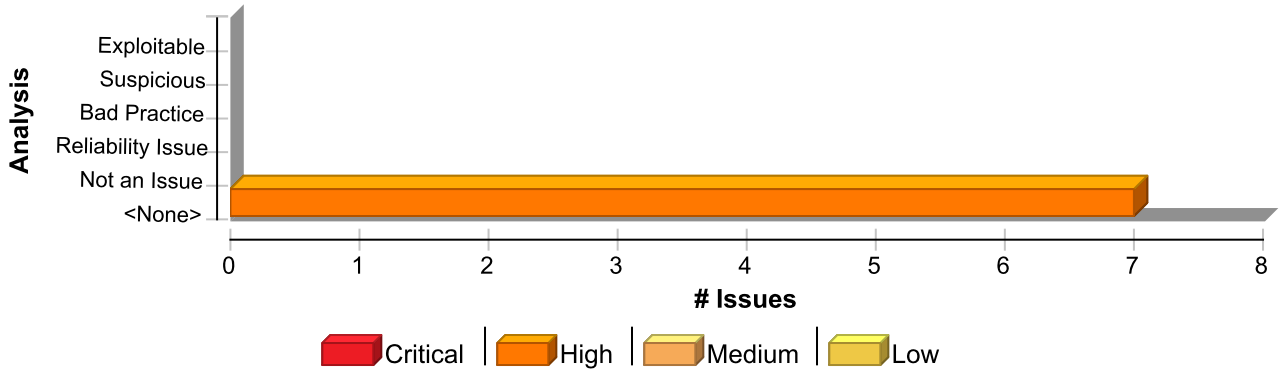
Explanation

The AccessibleObject API allows the programmer to get around the access control checks provided by Java access specifiers. In particular it enables the programmer to allow a reflected object to bypass Java access controls and in turn change the value of private fields or invoke private methods, behaviors that are normally disallowed.

Recommendation

Access specifiers should only be changed by a privileged class using arguments that an attacker cannot set. All occurrences should be examined carefully.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Access Specifier Manipulation	7	0	0	7
Total	**7**	**0**	**0**	**7**

Access Specifier Manipulation	**High**
Package: akka.io	
src/main/scala/akka/io/DirectByteBufferPool.scala, line 87 (Access Specifier Manipulation)	**High**

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: setAccessible()
Enclosing Method: liftedTree1()

Access Specifier Manipulation — High

Package: akka.io

src/main/scala/akka/io/DirectByteBufferPool.scala, line 87 (Access Specifier Manipulation) — High

File: src/main/scala/akka/io/DirectByteBufferPool.scala:87
Taint Flags:

```
84 cleanerMethod.setAccessible(true)
85
86 val cleanMethod = Class.forName("sun.misc.Cleaner").getMethod("clean")
87 cleanMethod.setAccessible(true)
88
89 { (bb: ByteBuffer) =>
90 try if (bb.isDirect) {
```

src/main/scala/akka/io/DirectByteBufferPool.scala, line 84 (Access Specifier Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: setAccessible()
Enclosing Method: liftedTree1()
File: src/main/scala/akka/io/DirectByteBufferPool.scala:84
Taint Flags:

```
81 private val CleanDirectBuffer: ByteBuffer => Unit =
82 try {
83 val cleanerMethod = Class.forName("java.nio.DirectByteBuffer").getMethod("cleaner")
84 cleanerMethod.setAccessible(true)
85
86 val cleanMethod = Class.forName("sun.misc.Cleaner").getMethod("clean")
87 cleanMethod.setAccessible(true)
```

Package: akka.util

src/main/scala/akka/util/LineNumbers.scala, line 202 (Access Specifier Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: setAccessible()
Enclosing Method: getStreamForLambda()
File: src/main/scala/akka/util/LineNumbers.scala:202
Taint Flags:

```
199 try {
200 val c = l.getClass
```

Access Specifier Manipulation — High

Package: akka.util

src/main/scala/akka/util/LineNumbers.scala, line 202 (Access Specifier Manipulation) — High

```
201 val writeReplace = c.getDeclaredMethod("writeReplace")
202 writeReplace.setAccessible(true)
203 writeReplace.invoke(l) match {
204 case serialized: SerializedLambda =>
205 if (debug)
```

src/main/scala/akka/util/Reflect.scala, line 72 (Access Specifier Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: setAccessible()
Enclosing Method: instantiate()
File: src/main/scala/akka/util/Reflect.scala:72
Taint Flags:

```
69 * Invokes the constructor with the given arguments.
70 */
71 private[akka] def instantiate[T](constructor: Constructor[T], args: immutable.Seq[Any]): T = {
72 constructor.setAccessible(true)
73 try constructor.newInstance(args.asInstanceOf[Seq[AnyRef]]: _*)
74 catch {
75 case e: IllegalArgumentException =>
```

src/main/scala/akka/util/Reflect.scala, line 54 (Access Specifier Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: setAccessible()
Enclosing Method: instantiate()
File: src/main/scala/akka/util/Reflect.scala:54
Taint Flags:

```
51 try ctor.newInstance()
52 catch {
53 case _: IllegalAccessException =>
54 ctor.setAccessible(true)
55 ctor.newInstance()
56 }
57 }
```

Access Specifier Manipulation — High

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/ReflectiveDynamicAccess.scala, line 39 (Access Specifier Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: setAccessible()
Enclosing Method: apply()
File: src/main/scala/akka/actor/ReflectiveDynamicAccess.scala:39
Taint Flags:

```
36 val types = args.map(_._1).toArray
37 val values = args.map(_._2).toArray
38 val constructor = clazz.getDeclaredConstructor(types: _*)
39 constructor.setAccessible(true)
40 val obj = constructor.newInstance(values: _*)
41 val t = implicitly[ClassTag[T]].runtimeClass
42 if (t.isInstance(obj)) obj.asInstanceOf[T]
```

src/main/scala/akka/actor/ReflectiveDynamicAccess.scala, line 66 (Access Specifier Manipulation) — High

Issue Details

Kingdom: Input Validation and Representation
Scan Engine: SCA (Semantic)

Sink Details

Sink: setAccessible()
Enclosing Method: apply()
File: src/main/scala/akka/actor/ReflectiveDynamicAccess.scala:66
Taint Flags:

```
63 classTry.flatMap { c =>
64 Try {
65 val module = c.getDeclaredField("MODULE$")
66 module.setAccessible(true)
67 val t = implicitly[ClassTag[T]].runtimeClass
68 module.get(null) match {
69 case null => throw new NullPointerException
```

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

Abstract

This class implements a `clone()` method but does not implement the `Cloneable` interface.

Explanation

It appears that the programmer intended for this class to implement the `Cloneable` interface because it implements a method named `clone()`. However, the class does not implement the `Cloneable` interface and the `clone()` method will not behave correctly. **Example 1:** Calling `clone()` for this class will result in a `CloneNotSupportedException.`

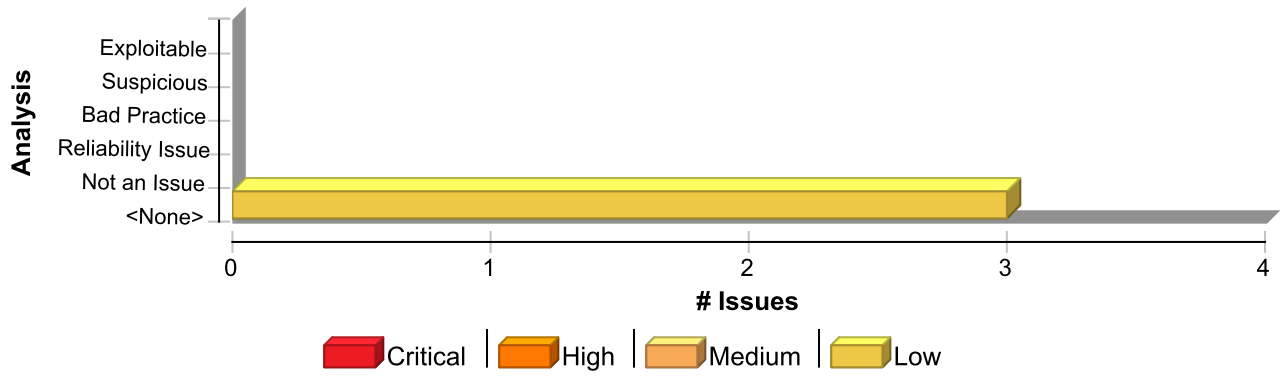
```
public class Kibitzer {
  public Object clone() throws CloneNotSupportedException {
    ...
  }
}
```

Recommendation

Implement both the `Cloneable` interface and the `clone()` method. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
public class Kibitzer implements Cloneable {
  public Object clone() throws CloneNotSupportedException {
    ...
  }
}
```

Issue Summary



Engine Breakdown

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

Code Correctness: Class Does Not Implement Cloneable — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 437 (Code Correctness: Class Does Not Implement Cloneable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: clone
Enclosing Method: clone()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:437
Taint Flags:

```
434 // *must* be overridden by derived classes. This construction is necessary
435 // to specialize the return type, as the method is already implemented in
436 // the parent class.
437 override def clone: ByteIterator =
438 throw new UnsupportedOperationException("Method clone is not implemented in ByteIterator")
439
440 override def duplicate: (ByteIterator, ByteIterator) = (this, clone)
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 259 (Code Correctness: Class Does Not Implement Cloneable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: clone
Enclosing Method: clone()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:259
Taint Flags:

```
256 case io => super.++(io)
257 }
258
259 final override def clone: MultiByteArrayIterator = {
260 val clonedIterators: List[ByteArrayIterator] = iterators.iterator.map(_.clone).to(List)
261 new MultiByteArrayIterator(clonedIterators)
262 }
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 72 (Code Correctness: Class Does Not Implement Cloneable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement Cloneable	Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 72 (Code Correctness: Class Does Not Implement Cloneable)	Low

Sink Details

Sink: Function: clone
Enclosing Method: clone()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:72
Taint Flags:

```
69 case io => super.++(io)
70 }
71
72 final override def clone: ByteArrayIterator = new ByteArrayIterator(array, from, until)
73
74 final override def take(n: Int): this.type = {
75 if (n < len) until = { if (n > 0) (from + n) else from }
```

Code Correctness: Class Does Not Implement equals (135 issues)

Abstract

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

Explanation

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

```
public class AccountGroup
{
    private int gid;

    public int getGid()
    {
        return gid;
    }

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

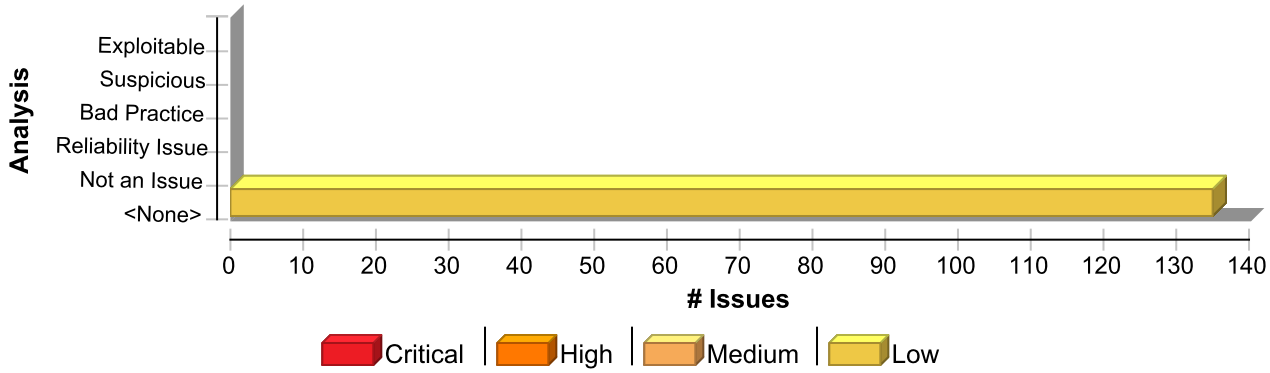
Recommendation

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

```
public class AccountGroup
{
    private int gid;

    public int getGid()
    {
        return gid;
    }

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

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

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Class Does Not Implement equals	135	0	0	135
Total	**135**	**0**	**0**	**135**

Code Correctness: Class Does Not Implement equals	**Low**
Package: akka	
src/main/scala/akka/AkkaVersion.scala, line 32 (Code Correctness: Class Does Not Implement equals)	**Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: require()
File: src/main/scala/akka/AkkaVersion.scala:32

Code Correctness: Class Does Not Implement equals — Low

Package: akka

src/main/scala/akka/AkkaVersion.scala, line 32 (Code Correctness: Class Does Not Implement equals) — Low

Taint Flags:

```
29 */
30 @InternalApi
31 private[akka] def require(libraryName: String, requiredVersion: String, currentVersion: String): Unit = {
32 if (requiredVersion != currentVersion) {
33 val VersionPattern = """(\d+)\.(\d+)\.(\d+)(-(?:M|RC)\d+)?""".r
34 currentVersion match {
35 case VersionPattern(currentMajorStr, currentMinorStr, currentPatchStr, mOrRc) =>
```

src/main/scala/akka/AkkaVersion.scala, line 44 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: require()
File: src/main/scala/akka/AkkaVersion.scala:44
Taint Flags:

```
41 else currentPatchStr.toInt
42 if (requiredMajorStr.toInt != currentMajorStr.toInt ||
43 requiredMinorStr.toInt > currentMinorStr.toInt ||
44 (requiredMinorStr == currentMinorStr && requiredPatchStr.toInt > currentPatch))
45 throw new UnsupportedAkkaVersion(
46 s"Current version of Akka is [$currentVersion], but $libraryName requires version [$requiredVersion]")
47 case _ => // SNAPSHOT or unknown - you're on your own
```

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 475 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: ActorSystem$Settings()
File: src/main/scala/akka/actor/ActorSystem.scala:475
Taint Flags:

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 475 (Code Correctness: Class Does Not Implement equals) — Low

```
472
473 final val DefaultVirtualNodesFactor: Int = getInt("akka.actor.deployment.default.virtual-nodes-factor")
474
475 if (ConfigVersion != Version)
476 throw new akka.ConfigurationException(
477 "Akka JAR version [" + Version + "] does not match the provided config version [" + ConfigVersion + "]")
478
```

src/main/scala/akka/actor/Deployer.scala, line 276 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: createRouterConfig()
File: src/main/scala/akka/actor/Deployer.scala:276
Taint Flags:

```
273 * @param deployment the deployment config, with defaults
274 */
275 protected def createRouterConfig(routerType: String, key: String, config: Config, deployment: Config): RouterConfig =
276 if (routerType == "from-code") NoRouter
277 else {
278 // need this for backwards compatibility, resizer enabled when including (parts of) resizer section in the deployment
279 val deployment2 =
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 362 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: executeTask()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:362
Taint Flags:

```
359 }
360
361 private[akka] final def executeTask(): Boolean = extractTask(ExecutedTask) match {
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 362 (Code Correctness: Class Does Not Implement equals) — Low

```
362 case ExecutedTask | CancelledTask => false
363 case other =>
364 try {
365 executionContext.execute(other)
```

src/main/scala/akka/actor/Deployer.scala, line 139 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: src/main/scala/akka/actor/Deployer.scala:139
Taint Flags:

```
136 override def equals(other: Any): Boolean = other match {
137 case that: Deploy =>
138 path == that.path &&
139 config == that.config &&
140 routerConfig == that.routerConfig &&
141 scope == that.scope &&
142 dispatcher == that.dispatcher &&
```

src/main/scala/akka/actor/ActorSystem.scala, line 916 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: stop()
File: src/main/scala/akka/actor/ActorSystem.scala:916
Taint Flags:

```
913 val sys = systemGuardian.path
914 path.parent match {
915 case `guard` => guardian ! StopChild(actor)
916 case `sys` => systemGuardian ! StopChild(actor)
917 case _ => actor.asInstanceOf[InternalActorRef].stop()
918 }
```

Code Correctness: Class Does Not Implement equals	Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 916 (Code Correctness: Class Does Not Implement equals) **Low**

```
919 }
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 357 (Code Correctness: Class Does Not Implement equals) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: extractTask()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:357
Taint Flags:

```
354 @tailrec
355 private final def extractTask(replaceWith: Runnable): Runnable =
356 task match {
357 case t @ (ExecutedTask | CancelledTask) => t
358 case x => if (unsafe.compareAndSwapObject(this, taskOffset, x, replaceWith)) x else extractTask(replaceWith)
359 }
360
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 362 (Code Correctness: Class Does Not Implement equals) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: executeTask()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:362
Taint Flags:

```
359 }
360
361 private[akka] final def executeTask(): Boolean = extractTask(ExecutedTask) match {
362 case ExecutedTask | CancelledTask => false
363 case other =>
364 try {
365 executionContext.execute(other)
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/AbstractProps.scala, line 62 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: applyOrElse()
File: src/main/scala/akka/actor/AbstractProps.scala:62
Taint Flags:

```
59 t.getActualTypeArguments.head match {
60 case c: Class[_] => c // since T <: Actor
61 case v: TypeVariable[_] =>
62 v.getBounds.collectFirst { case c: Class[_] if ac.isAssignableFrom(c) && c != ac => c }.getOrElse(ac)
63 case x => throw new IllegalArgumentException(s"unsupported type found in Creator argument [$x]")
64 }
65 case c: Class[_] if c == coc =>
```

src/main/scala/akka/actor/Deployer.scala, line 140 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: src/main/scala/akka/actor/Deployer.scala:140
Taint Flags:

```
137 case that: Deploy =>
138 path == that.path &&
139 config == that.config &&
140 routerConfig == that.routerConfig &&
141 scope == that.scope &&
142 dispatcher == that.dispatcher &&
143 mailbox == that.mailbox &&
```

src/main/scala/akka/actor/ActorSystem.scala, line 130 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 130 (Code Correctness: Class Does Not Implement equals) — Low

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/actor/ActorSystem.scala:130
Taint Flags:

```
127 case "local" => Local
128 // additional fqcn for older configs not using 'remote' or 'cluster'
129 case "remote" | RemoteActorRefProvider => Remote
130 case "cluster" | ClusterActorRefProvider => Cluster
131 case fqcn => Custom(fqcn)
132 }
133 }
```

src/main/scala/akka/actor/ActorSystem.scala, line 340 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: amendSlf4jConfig()
File: src/main/scala/akka/actor/ActorSystem.scala:340
Taint Flags:

```
337 val configuredLoggingFilter = config.getString(loggingFilterConfKey)
338
339 val loggingFilterAlreadyConfigured =
340 configuredLoggingFilter == slf4jLoggingFilterClassName || configuredLoggingFilter != classOf[
341 DefaultLoggingFilter].getName
342
343 def newLoggingFilterConfStr = s"""$loggingFilterConfKey = "$slf4jLoggingFilterClassName""""
```

src/main/scala/akka/actor/ActorRef.scala, line 549 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: isIgnoreRefPath()

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 549 (Code Correctness: Class Does Not Implement equals) — Low

File: src/main/scala/akka/actor/ActorRef.scala:549
Taint Flags:

```
546 * Check if the passed `otherPath` is the same as IgnoreActorRef.path
547 */
548 def isIgnoreRefPath(otherPath: ActorPath): Boolean =
549 path == otherPath
550
551 }
552
```

src/main/scala/akka/actor/FSM.scala, line 274 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: forMax()
File: src/main/scala/akka/actor/FSM.scala:274
Taint Flags:

```
271 */
272 def forMax(timeout: Duration): State[S, D] = timeout match {
273 case f: FiniteDuration => copy(timeout = Some(f))
274 case Duration.Inf => copy(timeout = SomeMaxFiniteDuration) // we map the Infinite duration to a special marker,
275 case _ => copy(timeout = None) // that means "cancel stateTimeout". This marker is needed
276 } // so we do not have to break source/binary compat.
277 // TODO: Can be removed once we can break State#timeout signature to `Option[Duration]`
```

src/main/scala/akka/actor/ActorPath.scala, line 446 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: rec()
File: src/main/scala/akka/actor/ActorPath.scala:446
Taint Flags:

```
443 def rec(left: ActorPath, right: ActorPath): Boolean =
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorPath.scala, line 446 (Code Correctness: Class Does Not Implement equals) — Low

```
444 if (left eq right) true
445 else if (left.isInstanceOf[RootActorPath]) left.equals(right)
446 else if (right.isInstanceOf[RootActorPath]) right.equals(left)
447 else left.name == right.name && rec(left.parent, right.parent)
448
449 other match {
```

src/main/scala/akka/actor/Deployer.scala, line 142 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: src/main/scala/akka/actor/Deployer.scala:142
Taint Flags:

```
139 config == that.config &&
140 routerConfig == that.routerConfig &&
141 scope == that.scope &&
142 dispatcher == that.dispatcher &&
143 mailbox == that.mailbox &&
144 tags == that.tags
145 case _ => false
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 688 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:688
Taint Flags:

```
685 }
686 case _ =>
687 // no deployment config found
688 if (props.deploy.dispatcher == Deploy.DispatcherSameAsParent)
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRefProvider.scala, line 688 (Code Correctness: Class Does Not Implement equals) — Low

```
689 props.withDispatcher(parentDispatcher)
690 else
691 props
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 357 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: extractTask()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:357
Taint Flags:

```
354 @tailrec
355 private final def extractTask(replaceWith: Runnable): Runnable =
356 task match {
357 case t @ (ExecutedTask | CancelledTask) => t
358 case x => if (unsafe.compareAndSwapObject(this, taskOffset, x, replaceWith)) x else extractTask(replaceWith)
359 }
360
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 557 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: guardian$lzycompute()
File: src/main/scala/akka/actor/ActorRefProvider.scala:557
Taint Flags:

```
554 case None => internalDispatcher
555 case Some(props) =>
556 val dispatcherId =
557 if (props.deploy.dispatcher == Deploy.DispatcherSameAsParent) Dispatchers.DefaultDispatcherId
558 else props.dispatcher
559 system.dispatchers.lookup(dispatcherId)
560 }
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSelection.scala, line 214 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: applyOrElse()
File: src/main/scala/akka/actor/ActorSelection.scala:214
Taint Flags:

```
211 .collect {
212 case x if x.nonEmpty =>
213 if ((x.indexOf('?') != -1) || (x.indexOf('*') != -1)) SelectChildPattern(x)
214 else if (x == "..") SelectParent
215 else SelectChildName(x)
216 }
217 .to(immutable.IndexedSeq)
```

src/main/scala/akka/actor/Deployer.scala, line 105 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: withFallback()
File: src/main/scala/akka/actor/Deployer.scala:105
Taint Flags:

```
102 config.withFallback(other.config),
103 routerConfig.withFallback(other.routerConfig),
104 scope.withFallback(other.scope),
105 if (dispatcher == Deploy.NoDispatcherGiven) other.dispatcher else dispatcher,
106 if (mailbox == Deploy.NoMailboxGiven) other.mailbox else mailbox)
107 }
108
```

src/main/scala/akka/actor/Deployer.scala, line 143 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/Deployer.scala, line 143 (Code Correctness: Class Does Not Implement equals) — Low

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: src/main/scala/akka/actor/Deployer.scala:143
Taint Flags:

```
140 routerConfig == that.routerConfig &&
141 scope == that.scope &&
142 dispatcher == that.dispatcher &&
143 mailbox == that.mailbox &&
144 tags == that.tags
145 case _ => false
146 }
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 377 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: cancel()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:377
Taint Flags:

```
374 override def run(): Unit = extractTask(ExecutedTask).run()
375
376 override def cancel(): Boolean = extractTask(CancelledTask) match {
377 case ExecutedTask | CancelledTask => false
378 case _ => true
379 }
380
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 377 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: cancel()

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 377 (Code Correctness: Class Does Not Implement equals) — Low

File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:377
Taint Flags:

```
374 override def run(): Unit = extractTask(ExecutedTask).run()
375
376 override def cancel(): Boolean = extractTask(CancelledTask) match {
377 case ExecutedTask | CancelledTask => false
378 case _ => true
379 }
380
```

src/main/scala/akka/actor/ActorSystem.scala, line 129 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/actor/ActorSystem.scala:129
Taint Flags:

```
126 providerClass match {
127 case "local" => Local
128 // additional fqcn for older configs not using 'remote' or 'cluster'
129 case "remote" | RemoteActorRefProvider => Remote
130 case "cluster" | ClusterActorRefProvider => Cluster
131 case fqcn => Custom(fqcn)
132 }
```

src/main/scala/akka/actor/ActorSystem.scala, line 340 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: amendSlf4jConfig()
File: src/main/scala/akka/actor/ActorSystem.scala:340
Taint Flags:

```
337 val configuredLoggingFilter = config.getString(loggingFilterConfKey)
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 340 (Code Correctness: Class Does Not Implement equals) — Low

```
338
339 val loggingFilterAlreadyConfigured =
340 configuredLoggingFilter == slf4jLoggingFilterClassName || configuredLoggingFilter != classOf[
341 DefaultLoggingFilter].getName
342
343 def newLoggingFilterConfStr = s"""$loggingFilterConfKey = "$slf4jLoggingFilterClassName""""
```

src/main/scala/akka/actor/TypedActor.scala, line 676 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: actorProps()
File: src/main/scala/akka/actor/TypedActor.scala:676
Taint Flags:

```
673 * Returns the akka.actor.Props representation of this TypedProps
674 */
675 def actorProps(): Props =
676 if (dispatcher == Props.default.dispatcher)
677 Props.default.withDeploy(deploy)
678 else Props.default.withDispatcher(dispatcher).withDeploy(deploy)
679 }
```

src/main/scala/akka/actor/ActorSystem.scala, line 915 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: stop()
File: src/main/scala/akka/actor/ActorSystem.scala:915
Taint Flags:

```
912 val guard = guardian.path
913 val sys = systemGuardian.path
914 path.parent match {
915 case `guard` => guardian ! StopChild(actor)
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 915 (Code Correctness: Class Does Not Implement equals) — Low

```
916 case `sys` => systemGuardian ! StopChild(actor)
917 case _ => actor.asInstanceOf[InternalActorRef].stop()
918 }
```

src/main/scala/akka/actor/Deployer.scala, line 141 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: src/main/scala/akka/actor/Deployer.scala:141
Taint Flags:

```
138 path == that.path &&
139 config == that.config &&
140 routerConfig == that.routerConfig &&
141 scope == that.scope &&
142 dispatcher == that.dispatcher &&
143 mailbox == that.mailbox &&
144 tags == that.tags
```

src/main/scala/akka/actor/ActorRef.scala, line 543 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: isIgnoreRefPath()
File: src/main/scala/akka/actor/ActorRef.scala:543
Taint Flags:

```
540 * Check if the passed `otherPath` is the same as IgnoreActorRef.path
541 */
542 def isIgnoreRefPath(otherPath: String): Boolean =
543 pathString == otherPath
544
545 /**
546 * Check if the passed `otherPath` is the same as IgnoreActorRef.path
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 422 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: run()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:422
Taint Flags:

```
419 case Running(otherJob) => Running(otherJob)
420 }
421 nextTaskState match {
422 case Running(runningJob) if runningJob == job =>
423 // only start the job if atomic update succeeds and we were the winner of any race
424 if (log.isDebugEnabled) {
425 log.debug("Performing task [{}] in CoordinatedShutdown phase [{}]", name, phaseName)
```

src/main/scala/akka/actor/ActorCell.scala, line 676 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: rootCauseOf()
File: src/main/scala/akka/actor/ActorCell.scala:676
Taint Flags:

```
673
674 @tailrec
675 private def rootCauseOf(throwable: Throwable): Throwable = {
676 if (throwable.getCause != null && throwable.getCause != throwable)
677 rootCauseOf(throwable.getCause)
678 else
679 throwable
```

src/main/scala/akka/actor/ActorPath.scala, line 447 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorPath.scala, line 447 (Code Correctness: Class Does Not Implement equals) — Low

Sink Details

Sink: FunctionCall: equals
Enclosing Method: rec()
File: src/main/scala/akka/actor/ActorPath.scala:447
Taint Flags:

```
444 if (left eq right) true
445 else if (left.isInstanceOf[RootActorPath]) left.equals(right)
446 else if (right.isInstanceOf[RootActorPath]) right.equals(left)
447 else left.name == right.name && rec(left.parent, right.parent)
448
449 other match {
450 case p: ActorPath => rec(this, p)
```

src/main/scala/akka/actor/AbstractProps.scala, line 65 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: create()
File: src/main/scala/akka/actor/AbstractProps.scala:65
Taint Flags:

```
62 v.getBounds.collectFirst { case c: Class[_] if ac.isAssignableFrom(c) && c != ac => c }.getOrElse(ac)
63 case x => throw new IllegalArgumentException(s"unsupported type found in Creator argument [$x]")
64 }
65 case c: Class[_] if c == coc =>
66 throw new IllegalArgumentException(
67 "erased Creator types (e.g. lambdas) are unsupported, use Props.create(actorClass, creator) instead")
68 case unexpected =>
```

src/main/scala/akka/actor/ActorPath.scala, line 445 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: rec()

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorPath.scala, line 445 (Code Correctness: Class Does Not Implement equals) — Low

File: src/main/scala/akka/actor/ActorPath.scala:445
Taint Flags:

```
442 @tailrec
443 def rec(left: ActorPath, right: ActorPath): Boolean =
444 if (left eq right) true
445 else if (left.isInstanceOf[RootActorPath]) left.equals(right)
446 else if (right.isInstanceOf[RootActorPath]) right.equals(left)
447 else left.name == right.name && rec(left.parent, right.parent)
448
```

src/main/scala/akka/actor/Deployer.scala, line 138 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: src/main/scala/akka/actor/Deployer.scala:138
Taint Flags:

```
135
136 override def equals(other: Any): Boolean = other match {
137 case that: Deploy =>
138 path == that.path &&
139 config == that.config &&
140 routerConfig == that.routerConfig &&
141 scope == that.scope &&
```

src/main/scala/akka/actor/Deployer.scala, line 106 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: withFallback()
File: src/main/scala/akka/actor/Deployer.scala:106
Taint Flags:

```
103 routerConfig.withFallback(other.routerConfig),
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/Deployer.scala, line 106 (Code Correctness: Class Does Not Implement equals) — Low

```
104 scope.withFallback(other.scope),
105 if (dispatcher == Deploy.NoDispatcherGiven) other.dispatcher else dispatcher,
106 if (mailbox == Deploy.NoMailboxGiven) other.mailbox else mailbox)
107 }
108
109 def withTags(tags: Set[String]): Deploy =
```

src/main/scala/akka/actor/Props.scala, line 175 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: withDispatcher()
File: src/main/scala/akka/actor/Props.scala:175
Taint Flags:

```
172 */
173 def withDispatcher(d: String): Props = deploy.dispatcher match {
174 case NoDispatcherGiven => copy(deploy = deploy.copy(dispatcher = d))
175 case x => if (x == d) this else copy(deploy = deploy.copy(dispatcher = d))
176 }
177
178 /**
```

src/main/scala/akka/actor/ActorRef.scala, line 174 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()
File: src/main/scala/akka/actor/ActorRef.scala:174
Taint Flags:

```
171 * Equals takes path and the unique id of the actor cell into account.
172 */
173 final override def equals(that: Any): Boolean = that match {
174 case other: ActorRef => path.uid == other.path.uid && path == other.path
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 174 (Code Correctness: Class Does Not Implement equals) — Low

```
175 case _ => false
176 }
177
```

src/main/scala/akka/actor/Props.scala, line 183 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: withMailbox()
File: src/main/scala/akka/actor/Props.scala:183
Taint Flags:

```
180 */
181 def withMailbox(m: String): Props = deploy.mailbox match {
182 case NoMailboxGiven => copy(deploy = deploy.copy(mailbox = m))
183 case x => if (x == m) this else copy(deploy = deploy.copy(mailbox = m))
184 }
185
186 /**
```

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/ChildrenContainer.scala, line 207 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: isTerminating()
File: src/main/scala/akka/actor/dungeon/ChildrenContainer.scala:207
Taint Flags:

```
204 case _ => this
205 }
206
207 override def isTerminating: Boolean = reason == Termination
208 override def isNormal: Boolean = reason == UserRequest
209
```

Code Correctness: Class Does Not Implement equals	Low

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/ChildrenContainer.scala, line 207 (Code Correctness: Class Does Not Implement equals) **Low**

```
210 override def toString =
```

src/main/scala/akka/actor/dungeon/ChildrenContainer.scala, line 208 (Code Correctness: Class Does Not Implement equals) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: isNormal()
File: src/main/scala/akka/actor/dungeon/ChildrenContainer.scala:208
Taint Flags:

```
205 }
206
207 override def isTerminating: Boolean = reason == Termination
208 override def isNormal: Boolean = reason == UserRequest
209
210 override def toString =
211 if (c.size > 20) c.size.toString + " children"
```

src/main/scala/akka/actor/dungeon/Children.scala, line 268 (Code Correctness: Class Does Not Implement equals) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: makeChild()
File: src/main/scala/akka/actor/dungeon/Children.scala:268
Taint Flags:

```
265 async: Boolean,
266 systemService: Boolean): ActorRef = {
267 val settings = cell.system.settings
268 if (settings.SerializeAllCreators && !systemService && props.deploy.scope != LocalScope) {
269 val oldInfo = Serialization.currentTransportInformation.value
270 try {
271 val ser = SerializationExtension(cell.system)
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/Children.scala, line 68 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: addFunctionRef()
File: src/main/scala/akka/actor/dungeon/Children.scala:68
Taint Flags:

```
65
66  private[akka] def addFunctionRef(f: (ActorRef, Any) => Unit, name: String = ""): FunctionRef = {
67    val r = randomName(new java.lang.StringBuilder("$$"))
68    val n = if (name != "") s"$r-$name" else r
69    val childPath = new ChildActorPath(self.path, n, ActorCell.newUid())
70    val ref = new FunctionRef(childPath, provider, system, f)
71
```

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailboxes.scala, line 160 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getMailboxType()
File: src/main/scala/akka/dispatch/Mailboxes.scala:160
Taint Flags:

```
157
158  val hasMailboxType =
159    dispatcherConfig.hasPath("mailbox-type") &&
160    dispatcherConfig.getString("mailbox-type") != Deploy.NoMailboxGiven
161
162  // TODO remove in 2.3
163  if (!hasMailboxType && !mailboxSizeWarningIssued && dispatcherConfig.hasPath("mailbox-size")) {
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 185 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse

Code Correctness: Class Does Not Implement equals — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailboxes.scala, line 185 (Code Correctness: Class Does Not Implement equals) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getMailboxType()
File: src/main/scala/akka/dispatch/Mailboxes.scala:185
Taint Flags:

```
182  mailboxType
183  }
184
185  if (deploy.mailbox != Deploy.NoMailboxGiven) {
186  verifyRequirements(lookup(deploy.mailbox))
187  } else if (deploy.dispatcher != Deploy.NoDispatcherGiven && deploy.dispatcher != Deploy.DispatcherSameAsParent &&
hasMailboxType) {
188  verifyRequirements(lookup(dispatcherConfig.getString("id")))
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 187 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getMailboxType()
File: src/main/scala/akka/dispatch/Mailboxes.scala:187
Taint Flags:

```
184
185  if (deploy.mailbox != Deploy.NoMailboxGiven) {
186  verifyRequirements(lookup(deploy.mailbox))
187  } else if (deploy.dispatcher != Deploy.NoDispatcherGiven && deploy.dispatcher != Deploy.DispatcherSameAsParent &&
hasMailboxType) {
188  verifyRequirements(lookup(dispatcherConfig.getString("id")))
189  } else if (hasRequiredType(actorClass)) {
190  try verifyRequirements(lookupByQueueType(getRequiredType(actorClass)))
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 421 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement equals — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 421 (Code Correctness: Class Does Not Implement equals) — Low

Sink Details

Sink: FunctionCall: equals
Enclosing Method: createThreadPoolConfigBuilder()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:421
Taint Flags:

```
418 case _ => None
419 })
420
421 if (config.getString("fixed-pool-size") == "off")
422 builder
423 .setCorePoolSizeFromFactor(
424 config.getInt("core-pool-size-min"),
```

src/main/scala/akka/dispatch/Mailbox.scala, line 490 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: systemDrain()
File: src/main/scala/akka/dispatch/Mailbox.scala:490
Taint Flags:

```
487 @tailrec
488 final def systemDrain(newContents: LatestFirstSystemMessageList): EarliestFirstSystemMessageList = {
489 val currentList = systemQueueGet
490 if (currentList.head == NoMessage) new EarliestFirstSystemMessageList(null)
491 else if (systemQueuePut(currentList, newContents)) currentList.reverse
492 else systemDrain(newContents)
493 }
```

src/main/scala/akka/dispatch/Mailbox.scala, line 477 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: systemEnqueue()

Code Correctness: Class Does Not Implement equals — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 477 (Code Correctness: Class Does Not Implement equals) — Low

File: src/main/scala/akka/dispatch/Mailbox.scala:477
Taint Flags:

```
474 assert(message.unlinked)
475 if (Mailbox.debug) println("" + receiver + " having enqueued " + message)
476 val currentList = systemQueueGet
477 if (currentList.head == NoMessage) {
478 if (actor ne null) actor.dispatcher.mailboxes.deadLetterMailbox.systemEnqueue(receiver, message)
479 } else {
480 if (!systemQueuePut(currentList, message :: currentList)) {
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 298 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: stashCapacity()
File: src/main/scala/akka/dispatch/Mailboxes.scala:298
Taint Flags:

```
295 updateCache(stashCapacityCache.get, key, value) // recursive, try again
296 }
297
298 if (dispatcher == Dispatchers.DefaultDispatcherId && mailbox == Mailboxes.DefaultMailboxId)
299 defaultStashCapacity
300 else {
301 val cache = stashCapacityCache.get
```

src/main/scala/akka/dispatch/Dispatchers.scala, line 311 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: amendConfig()
File: src/main/scala/akka/dispatch/Dispatchers.scala:311
Taint Flags:

```
308 private val defaultRequirement =
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Dispatchers.scala, line 311 (Code Correctness: Class Does Not Implement equals) — Low

```
309 ConfigFactory.parseString("mailbox-requirement = akka.dispatch.MultipleConsumerSemantics")
310 def amendConfig(config: Config): Config =
311 if (config.getString("mailbox-requirement") != Mailboxes.NoMailboxRequirement) config
312 else defaultRequirement.withFallback(config)
313 }
314
```

src/main/scala/akka/dispatch/CachingConfig.scala, line 62 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getPathEntry()
File: src/main/scala/akka/dispatch/CachingConfig.scala:62
Taint Flags:

```
59 case Failure(_) =>
60 emptyPathEntry
61 case Success(v) =>
62 if (v.valueType() == ConfigValueType.STRING)
63 StringPathEntry(true, true, v.atKey("cached"), v.unwrapped().asInstanceOf[String])
64 else
65 ValuePathEntry(true, true, v.atKey("cached"))
```

src/main/scala/akka/dispatch/PinnedDispatcher.scala, line 38 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: register()
File: src/main/scala/akka/dispatch/PinnedDispatcher.scala:38
Taint Flags:

```
35 //Relies on an external lock provided by MessageDispatcher.attach
36 protected[akka] override def register(actorCell: ActorCell) = {
37 val actor = owner
38 if ((actor ne null) && actorCell != actor)
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/PinnedDispatcher.scala, line 38 (Code Correctness: Class Does Not Implement equals) — Low

```
39 throw new IllegalArgumentException("Cannot register to anyone but " + actor)
40 owner = actorCell
41 super.register(actorCell)
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 187 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getMailboxType()
File: src/main/scala/akka/dispatch/Mailboxes.scala:187
Taint Flags:

```
184
185 if (deploy.mailbox != Deploy.NoMailboxGiven) {
186 verifyRequirements(lookup(deploy.mailbox))
187 } else if (deploy.dispatcher != Deploy.NoDispatcherGiven && deploy.dispatcher != Deploy.DispatcherSameAsParent && hasMailboxType) {
188 verifyRequirements(lookup(dispatcherConfig.getString("id")))
189 } else if (hasRequiredType(actorClass)) {
190 try verifyRequirements(lookupByQueueType(getRequiredType(actorClass)))
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 298 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: stashCapacity()
File: src/main/scala/akka/dispatch/Mailboxes.scala:298
Taint Flags:

```
295 updateCache(stashCapacityCache.get, key, value) // recursive, try again
296 }
297
298 if (dispatcher == Dispatchers.DefaultDispatcherId && mailbox == Mailboxes.DefaultMailboxId)
299 defaultStashCapacity
300 else {
301 val cache = stashCapacityCache.get
```

Code Correctness: Class Does Not Implement equals	Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailboxes.scala, line 298 (Code Correctness: Class Does Not Implement equals)	Low

src/main/scala/akka/dispatch/Mailboxes.scala, line 317 (Code Correctness: Class Does Not Implement equals)	Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: stashCapacityFromConfig()
File: src/main/scala/akka/dispatch/Mailboxes.scala:317
Taint Flags:

```
314 val disp = Dispatchers.getConfig(settings.config, dispatcher)
315 val fallback = disp.withFallback(settings.config.getConfig(Mailboxes.DefaultMailboxId))
316 val config =
317 if (mailbox == Mailboxes.DefaultMailboxId) fallback
318 else settings.config.getConfig(mailbox).withFallback(fallback)
319 config.getInt("stash-capacity")
320 }
```

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 999 (Code Correctness: Class Does Not Implement equals)	Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: error()
File: src/main/scala/akka/event/Logging.scala:999
Taint Flags:

```
996
997 def error(event: Error): Unit = event match {
998 case e: Error3 => // has marker
999 val f = if (event.cause == Error.NoCause) ErrorWithoutCauseWithMarkerFormat else ErrorFormatWithMarker
1000 println(
1001 f.format(
1002 e.marker.name,
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 1010 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: error()
File: src/main/scala/akka/event/Logging.scala:1010
Taint Flags:

```
1007  event.message,
1008  stackTraceFor(event.cause)))
1009  case _ =>
1010  val f = if (event.cause == Error.NoCause) ErrorFormatWithoutCause else ErrorFormat
1011  println(
1012  f.format(
1013  timestamp(event),
```

Package: akka.io

src/main/scala/akka/io/TcpConnection.scala, line 386 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: stopWith()
File: src/main/scala/akka/io/TcpConnection.scala:386
Taint Flags:

```
383  closedMessage = Some(closeInfo)
384  unsignDeathPact()
385
386  if (closeInfo.closedEvent == Aborted || shouldAbort)
387  prepareAbort()
388
389  registration match {
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 136 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse

Code Correctness: Class Does Not Implement equals — Low

Package: akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 136 (Code Correctness: Class Does Not Implement equals) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: applyOrElse()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:136
Taint Flags:

```
133 } catch {
134 case _: UnknownHostException =>
135 val answer = DnsProtocol.Resolved(name, immutable.Seq.empty)
136 if (negativeCachePolicy != Never)
137 cache.put((name, ip), answer, negativeCachePolicy)
138 answer
139 }
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 159 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: applyOrElse()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:159
Taint Flags:

```
156 } catch {
157 case _: UnknownHostException =>
158 val answer = Dns.Resolved(name, immutable.Seq.empty, immutable.Seq.empty)
159 if (negativeCachePolicy != Never)
160 cache.put((name, Ip()), DnsProtocol.Resolved(name, immutable.Seq.empty), negativeCachePolicy)
161 answer
162 }
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 130 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Class Does Not Implement equals — Low

Package: akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 130 (Code Correctness: Class Does Not Implement equals) — Low

Sink: FunctionCall: equals
Enclosing Method: applyOrElse()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:130
Taint Flags:

```
127 val addresses: Array[InetAddress] = InetAddress.getAllByName(name)
128 val records = addressToRecords(name, addresses.toList, ipv4, ipv6)
129 val answer = DnsProtocol.Resolved(name, records.toList)
130 if (positiveCachePolicy != Never)
131 cache.put((name, Ip()), DnsProtocol.Resolved(name, records), positiveCachePolicy)
132 answer
133 } catch {
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 151 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: applyOrElse()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:151
Taint Flags:

```
148 val addresses = InetAddress.getAllByName(name)
149 // respond with the old protocol as the request was the new protocol
150 val answer = Dns.Resolved(name, addresses)
151 if (positiveCachePolicy != Never) {
152 val records = addressToRecords(name, addresses.toList, ipv4 = true, ipv6 = true)
153 cache.put((name, Ip()), DnsProtocol.Resolved(name, records), positiveCachePolicy)
154 }
```

Package: akka.io.dns

src/main/scala/akka/io/dns/CachePolicy.scala, line 25 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: equals()

Code Correctness: Class Does Not Implement equals — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/CachePolicy.scala, line 25 (Code Correctness: Class Does Not Implement equals) — Low

File: src/main/scala/akka/io/dns/CachePolicy.scala:25
Taint Flags:

```
22 def getValue: java.time.Duration = value.asJava
23
24 override def equals(other: Any): Boolean = other match {
25 case that: Ttl => value == that.value
26 case _ => false
27 }
28
```

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/TcpDnsClient.scala, line 83 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: akkaiodns$internal$TcpDnsClient$$parseResponse()
File: src/main/scala/akka/io/dns/internal/TcpDnsClient.scala:83
Taint Flags:

```
80 log.warning("TCP DNS response truncated")
81 }
82 val (recs, additionalRecs) =
83 if (msg.flags.responseCode == ResponseCode.SUCCESS) (msg.answerRecs, msg.additionalRecs) else (Nil, Nil)
84 Answer(msg.id, recs, additionalRecs)
85 }
86 }
```

src/main/scala/akka/io/dns/internal/DnsClient.scala, line 134 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: applyOrElse()
File: src/main/scala/akka/io/dns/internal/DnsClient.scala:134
Taint Flags:

Code Correctness: Class Does Not Implement equals	Low

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsClient.scala, line 134 (Code Correctness: Class Does Not Implement equals) **Low**

```
131 }
132 } else {
133 val (recs, additionalRecs) =
134 if (msg.flags.responseCode == ResponseCode.SUCCESS) (msg.answerRecs, msg.additionalRecs) else (Nil, Nil)
135 self ! Answer(msg.id, recs, additionalRecs)
136 }
137 case response: Answer =>
```

Package: akka.pattern

src/main/scala/akka/pattern/CircuitBreaker.scala, line 471 (Code Correctness: Class Does Not Implement equals) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: isOpen()
File: src/main/scala/akka/pattern/CircuitBreaker.scala:471
Taint Flags:

```
468 * manage the state yourself.
469 */
470 def isOpen: Boolean = {
471 currentState == Open
472 }
473
474 /**
```

src/main/scala/akka/pattern/CircuitBreaker.scala, line 481 (Code Correctness: Class Does Not Implement equals) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: isHalfOpen()
File: src/main/scala/akka/pattern/CircuitBreaker.scala:481
Taint Flags:

```
478 * manage the state yourself.
479 */
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.pattern

src/main/scala/akka/pattern/CircuitBreaker.scala, line 481 (Code Correctness: Class Does Not Implement equals) — Low

```
480 def isHalfOpen: Boolean = {
481 currentState == HalfOpen
482 }
483
484 /**
```

src/main/scala/akka/pattern/CircuitBreaker.scala, line 461 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: isClosed()
File: src/main/scala/akka/pattern/CircuitBreaker.scala:461
Taint Flags:

```
458 * manage the state yourself.
459 */
460 def isClosed: Boolean = {
461 currentState == Closed
462 }
463
464 /**
```

src/main/scala/akka/pattern/CircuitBreaker.scala, line 794 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: callThrough()
File: src/main/scala/akka/pattern/CircuitBreaker.scala:794
Taint Flags:

```
791 try value
792 catch { case NonFatal(t) => Future.failed(t) }
793
794 if (callTimeout == Duration.Zero) {
795 val start = System.nanoTime()
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.pattern

src/main/scala/akka/pattern/CircuitBreaker.scala, line 794 (Code Correctness: Class Does Not Implement equals) — Low

```
796 val f = materialize(body)
797
```

Package: akka.routing

src/main/scala/akka/routing/Balancing.scala, line 141 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: withFallback()
File: src/main/scala/akka/routing/Balancing.scala:141
Taint Flags:

```
138 * if this RouterConfig doesn't have one.
139 */
140 override def withFallback(other: RouterConfig): RouterConfig =
141 if (other == NoRouter) this // NoRouter is the default, hence “neutral”
142 else {
143
144 other match {
```

src/main/scala/akka/routing/RoutedActorRef.scala, line 39 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: RoutedActorRef()
File: src/main/scala/akka/routing/RoutedActorRef.scala:39
Taint Flags:

```
36 extends RepointableActorRef(_system, _routerProps, _routerDispatcher, _routerMailbox, _supervisor, _path) {
37
38 // verify that a BalancingDispatcher is not used with a Router
39 if (_routerProps.routerConfig != NoRouter && _routerDispatcher.isInstanceOf[BalancingDispatcher]) {
40 throw new ConfigurationException(
41 "Configuration for " + this +
42 " is invalid - you can not use a 'BalancingDispatcher' as a Router's dispatcher, you can however use it for the routees.")
```

Code Correctness: Class Does Not Implement equals	Low

Package: akka.routing

src/main/scala/akka/routing/RoutedActorRef.scala, line 39 (Code Correctness: Class Does Not Implement equals)	Low

src/main/scala/akka/routing/RouterConfig.scala, line 210 (Code Correctness: Class Does Not Implement equals)	Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: enrichWithPoolDispatcher()
File: src/main/scala/akka/routing/RouterConfig.scala:210
Taint Flags:

```
207 * INTERNAL API
208 */
209 private[akka] def enrichWithPoolDispatcher(routeeProps: Props, context: ActorContext): Props =
210 if (usePoolDispatcher && routeeProps.dispatcher == Dispatchers.DefaultDispatcherId)
211 routeeProps.withDispatcher(
212 "akka.actor.deployment." + context.self.path.elements.drop(1).mkString("/", "/", "")
213 + ".pool-dispatcher")
```

src/main/scala/akka/routing/Router.scala, line 125 (Code Correctness: Class Does Not Implement equals)	Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: send()
File: src/main/scala/akka/routing/Router.scala:125
Taint Flags:

```
122 }
123
124 private def send(routee: Routee, msg: Any, sender: ActorRef): Unit = {
125 if (routee == NoRoutee && sender.isInstanceOf[InternalActorRef])
126 sender.asInstanceOf[InternalActorRef].provider.deadLetters.tell(unwrap(msg), sender)
127 else
128 routee.send(unwrap(msg), sender)
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.routing

src/main/scala/akka/routing/RouterConfig.scala, line 109 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: overrideUnsetConfig()
File: src/main/scala/akka/routing/RouterConfig.scala:109
Taint Flags:

```
106 private[akka] trait PoolOverrideUnsetConfig[T <: Pool] extends Pool {
107
108  final def overrideUnsetConfig(other: RouterConfig): RouterConfig =
109  if (other == NoRouter) this // NoRouter is the default, hence “neutral”
110  else {
111
112  other match {
```

Package: akka.serialization

src/main/scala/akka/serialization/PrimitiveSerializers.scala, line 186 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: toBinary()
File: src/main/scala/akka/serialization/PrimitiveSerializers.scala:186
Taint Flags:

```
183
184  override def toBinary(o: AnyRef): Array[Byte] = {
185  val flag = o match {
186  case TRUE => TrueB
187  case FALSE => FalseB
188  case b => throw new IllegalArgumentException(s"Non boolean flag: $b")
189  }
```

src/main/scala/akka/serialization/PrimitiveSerializers.scala, line 170 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse

Code Correctness: Class Does Not Implement equals — Low

Package: akka.serialization

src/main/scala/akka/serialization/PrimitiveSerializers.scala, line 170 (Code Correctness: Class Does Not Implement equals) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: toBinary()
File: src/main/scala/akka/serialization/PrimitiveSerializers.scala:170
Taint Flags:

```
167 override def toBinary(o: AnyRef, buf: ByteBuffer): Unit = {
168 val flag = o match {
169 case TRUE => TrueB
170 case FALSE => FalseB
171 case b => throw new IllegalArgumentException(s"Non boolean flag: $b")
172 }
173 buf.put(flag)
```

src/main/scala/akka/serialization/PrimitiveSerializers.scala, line 169 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: toBinary()
File: src/main/scala/akka/serialization/PrimitiveSerializers.scala:169
Taint Flags:

```
166
167 override def toBinary(o: AnyRef, buf: ByteBuffer): Unit = {
168 val flag = o match {
169 case TRUE => TrueB
170 case FALSE => FalseB
171 case b => throw new IllegalArgumentException(s"Non boolean flag: $b")
172 }
```

src/main/scala/akka/serialization/Serialization.scala, line 76 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Class Does Not Implement equals — Low

Package: akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 76 (Code Correctness: Class Does Not Implement equals) — Low

Sink: FunctionCall: equals
Enclosing Method: serializedActorPath()
File: src/main/scala/akka/serialization/Serialization.scala:76
Taint Flags:

```
73 catch { case NonFatal(_) => path.toSerializationFormat }
74 }
75 case Information(address, system) =>
76 if (originalSystem == null || originalSystem == system)
77 path.toSerializationFormatWithAddress(address)
78 else {
79 val provider = originalSystem.provider
```

src/main/scala/akka/serialization/PrimitiveSerializers.scala, line 187 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: toBinary()
File: src/main/scala/akka/serialization/PrimitiveSerializers.scala:187
Taint Flags:

```
184 override def toBinary(o: AnyRef): Array[Byte] = {
185 val flag = o match {
186 case TRUE => TrueB
187 case FALSE => FalseB
188 case b => throw new IllegalArgumentException(s"Non boolean flag: $b")
189 }
190 val result = new Array[Byte](1)
```

src/main/scala/akka/serialization/Serialization.scala, line 383 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: serializerOf()
File: src/main/scala/akka/serialization/Serialization.scala:383
Taint Flags:

Code Correctness: Class Does Not Implement equals — Low

Package: akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 383 (Code Correctness: Class Does Not Implement equals) — Low

```
380 private def serializerOf(bindingName: String, serializerFQN: String): Try[Serializer] = {
381 // We override each instantiation of the JavaSerializer with the "disabled" serializer which will log warnings if used.
382 val fqn =
383 if (!system.settings.AllowJavaSerialization && serializerFQN == classOf[JavaSerializer].getName) {
384 log.debug(
385 "Replacing JavaSerializer with DisabledJavaSerializer, " +
386 "due to `akka.actor.allow-java-serialization = off`.")
```

src/main/scala/akka/serialization/Serialization.scala, line 394 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: applyOrElse()
File: src/main/scala/akka/serialization/Serialization.scala:394
Taint Flags:

```
391 case _: NoSuchMethodException =>
392 system.dynamicAccess.createInstanceFor[Serializer](fqn, Nil).recoverWith {
393 case e: NoSuchMethodException =>
394 if (bindingName == "") throw e // compatibility with (public) serializerOf method without bindingName
395 else
396 system.dynamicAccess.createInstanceFor[Serializer](
397 fqn,
```

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 557 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getLong()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:557
Taint Flags:

```
554 * Get a single Long from this iterator.
555 */
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 557 (Code Correctness: Class Does Not Implement equals) — Low

```
556 def getLong(implicit byteOrder: ByteOrder): Long = {
557 if (byteOrder == ByteOrder.BIG_ENDIAN)
558 ((next().toLong & 0xff) << 56
559 | (next().toLong & 0xff) << 48
560 | (next().toLong & 0xff) << 40
```

src/main/scala/akka/util/WildcardIndex.scala, line 60 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: insert()
File: src/main/scala/akka/util/WildcardIndex.scala:60
Taint Flags:

```
57 if (e != "**" && e.endsWith("**"))
58 throw new IllegalArgumentException(
59 "double wildcard can't be used as a suffix (e.g. /user/actor**), only as a full subPath element (e.g. /user/actor/**)")
60 else if (e != "*" && e != "**" && e.endsWith("*"))
61 copy(
62 wildcardSuffixChildren = wildcardSuffixChildren
63 .updated(e.stripSuffix("*"), wildcardSuffixChildren.getOrElse(e, WildcardTree[T]()).insert(elems, d)))
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 531 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getShort()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:531
Taint Flags:

```
528 def getShort(implicit byteOrder: ByteOrder): Short = {
529 if (byteOrder == ByteOrder.BIG_ENDIAN)
530 ((next() & 0xff) << 8 | (next() & 0xff) << 0).toShort
531 else if (byteOrder == ByteOrder.LITTLE_ENDIAN)
532 ((next() & 0xff) << 0 | (next() & 0xff) << 8).toShort
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 531 (Code Correctness: Class Does Not Implement equals) — Low

```
533  else throw new IllegalArgumentException("Unknown byte order " + byteOrder)
534  }
```

src/main/scala-2.13/akka/util/ByteString.scala, line 232 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: indexOf()
File: src/main/scala-2.13/akka/util/ByteString.scala:232
Taint Flags:

```
229  var found = -1
230  var i = math.max(from, 0)
231  while (i < length && found == -1) {
232  if (bytes(i) == elem) found = i
233  i += 1
234  }
235  found
```

src/main/scala-2.13/akka/util/ByteString.scala, line 1212 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: putShort()
File: src/main/scala-2.13/akka/util/ByteString.scala:1212
Taint Flags:

```
1209  * Add a single Short to this builder.
1210  */
1211  def putShort(x: Int)(implicit byteOrder: ByteOrder): this.type = {
1212  if (byteOrder == ByteOrder.BIG_ENDIAN) {
1213  this += (x >>> 8).toByte
1214  this += (x >>> 0).toByte
1215  } else if (byteOrder == ByteOrder.LITTLE_ENDIAN) {
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 583 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getLongPart()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:583
Taint Flags:

```
580 * bytes were encoded.
581 */
582 def getLongPart(n: Int)(implicit byteOrder: ByteOrder): Long = {
583 if (byteOrder == ByteOrder.BIG_ENDIAN) {
584 var x = 0L
585 (1 to n).foreach(_ => x = (x << 8) | (next() & 0xff))
586 x
```

src/main/scala-2.13/akka/util/ByteString.scala, line 1215 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: putShort()
File: src/main/scala-2.13/akka/util/ByteString.scala:1215
Taint Flags:

```
1212 if (byteOrder == ByteOrder.BIG_ENDIAN) {
1213 this += (x >>> 8).toByte
1214 this += (x >>> 0).toByte
1215 } else if (byteOrder == ByteOrder.LITTLE_ENDIAN) {
1216 this += (x >>> 0).toByte
1217 this += (x >>> 8).toByte
1218 } else throw new IllegalArgumentException("Unknown byte order " + byteOrder)
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 545 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 545 (Code Correctness: Class Does Not Implement equals) — Low

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getInt()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:545
Taint Flags:

```
542 | (next() & 0xff) << 16
543 | (next() & 0xff) << 8
544 | (next() & 0xff) << 0)
545 else if (byteOrder == ByteOrder.LITTLE_ENDIAN)
546 ((next() & 0xff) << 0
547 | (next() & 0xff) << 8
548 | (next() & 0xff) << 16
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 587 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getLongPart()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:587
Taint Flags:

```
584 var x = 0L
585 (1 to n).foreach(_ => x = (x << 8) | (next() & 0xff))
586 x
587 } else if (byteOrder == ByteOrder.LITTLE_ENDIAN) {
588 var x = 0L
589 (0 until n).foreach(i => x |= (next() & 0xff) << 8 * i)
590 x
```

src/main/scala/akka/util/Reflect.scala, line 137 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: rec()

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala/akka/util/Reflect.scala, line 137 (Code Correctness: Class Does Not Implement equals) — Low

File: src/main/scala/akka/util/Reflect.scala:137
Taint Flags:

```
134 } match {
135 case None => throw new IllegalArgumentException(s"cannot find [$marker] in ancestors of [$root]")
136 case Some(c: Class[_]) => if (c == marker) c else rec(c)
137 case Some(t: ParameterizedType) => if (t.getRawType == marker) t else rec(t.getRawType.asInstanceOf[Class[_]])
138 case _ => ??? // cannot happen due to collectFirst
139 }
140 }
```

src/main/scala/akka/util/Version.scala, line 161 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: compareTo()
File: src/main/scala/akka/util/Version.scala:161
Taint Flags:

```
158 if (diff == 0) {
159 diff = numbers(3) - other.numbers(3)
160 if (diff == 0) {
161 if (rest == "" && other.rest != "")
162 diff = 1
163 if (other.rest == "" && rest != "")
164 diff = -1
```

src/main/scala/akka/util/Version.scala, line 163 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: compareTo()
File: src/main/scala/akka/util/Version.scala:163
Taint Flags:

```
160 if (diff == 0) {
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala/akka/util/Version.scala, line 163 (Code Correctness: Class Does Not Implement equals) — Low

```
161 if (rest == "" && other.rest != "")
162 diff = 1
163 if (other.rest == "" && rest != "")
164 diff = -1
165 else
166 diff = rest.compareTo(other.rest)
```

src/main/scala/akka/util/WildcardIndex.scala, line 57 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: insert()
File: src/main/scala/akka/util/WildcardIndex.scala:57
Taint Flags:

```
54 copy(data = Some(d))
55 } else {
56 val e = elems.next()
57 if (e != "**" && e.endsWith("**"))
58 throw new IllegalArgumentException(
59 "double wildcard can't be used as a suffix (e.g. /user/actor**), only as a full subPath element (e.g. /user/actor/**)")
60 else if (e != "*" && e != "**" && e.endsWith("*"))
```

src/main/scala/akka/util/WildcardIndex.scala, line 60 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: insert()
File: src/main/scala/akka/util/WildcardIndex.scala:60
Taint Flags:

```
57 if (e != "**" && e.endsWith("**"))
58 throw new IllegalArgumentException(
59 "double wildcard can't be used as a suffix (e.g. /user/actor**), only as a full subPath element (e.g. /user/actor/**)")
60 else if (e != "*" && e != "**" && e.endsWith("*"))
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala/akka/util/WildcardIndex.scala, line 60 (Code Correctness: Class Does Not Implement equals) — Low

```
61 copy(
62 wildcardSuffixChildren = wildcardSuffixChildren
63 .updated(e.stripSuffix("*"), wildcardSuffixChildren.getOrElse(e, WildcardTree[T]()).insert(elems, d)))
```

src/main/scala/akka/util/Version.scala, line 161 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: compareTo()
File: src/main/scala/akka/util/Version.scala:161
Taint Flags:

```
158 if (diff == 0) {
159 diff = numbers(3) - other.numbers(3)
160 if (diff == 0) {
161 if (rest == "" && other.rest != "")
162 diff = 1
163 if (other.rest == "" && rest != "")
164 diff = -1
```

src/main/scala/akka/util/Version.scala, line 163 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: compareTo()
File: src/main/scala/akka/util/Version.scala:163
Taint Flags:

```
160 if (diff == 0) {
161 if (rest == "" && other.rest != "")
162 diff = 1
163 if (other.rest == "" && rest != "")
164 diff = -1
165 else
166 diff = rest.compareTo(other.rest)
```

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 529 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getShort()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:529
Taint Flags:

```
526 * Get a single Short from this iterator.
527 */
528 def getShort(implicit byteOrder: ByteOrder): Short = {
529 if (byteOrder == ByteOrder.BIG_ENDIAN)
530 ((next() & 0xff) << 8 | (next() & 0xff) << 0).toShort
531 else if (byteOrder == ByteOrder.LITTLE_ENDIAN)
532 ((next() & 0xff) << 0 | (next() & 0xff) << 8).toShort
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 540 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getInt()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:540
Taint Flags:

```
537 * Get a single Int from this iterator.
538 */
539 def getInt(implicit byteOrder: ByteOrder): Int = {
540 if (byteOrder == ByteOrder.BIG_ENDIAN)
541 ((next() & 0xff) << 24
542 | (next() & 0xff) << 16
543 | (next() & 0xff) << 8
```

src/main/scala-2.13/akka/util/ByteString.scala, line 422 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 422 (Code Correctness: Class Does Not Implement equals) — Low

Sink Details

Sink: FunctionCall: equals
Enclosing Method: indexOf()
File: src/main/scala-2.13/akka/util/ByteString.scala:422
Taint Flags:

```
419 var found = -1
420 var i = math.max(from, 0)
421 while (i < length && found == -1) {
422 if (bytes(startIndex + i) == elem) found = i
423 i += 1
424 }
425 found
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 566 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: getLong()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:566
Taint Flags:

```
563 | (next().toLong & 0xff) << 16
564 | (next().toLong & 0xff) << 8
565 | (next().toLong & 0xff) << 0)
566 else if (byteOrder == ByteOrder.LITTLE_ENDIAN)
567 ((next().toLong & 0xff) << 0
568 | (next().toLong & 0xff) << 8
569 | (next().toLong & 0xff) << 16
```

src/main/scala/akka/util/Version.scala, line 147 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: compareTo()

Code Correctness: Class Does Not Implement equals — Low

Package: akka.util

src/main/scala/akka/util/Version.scala, line 147 (Code Correctness: Class Does Not Implement equals) — Low

File: src/main/scala/akka/util/Version.scala:147
Taint Flags:

```
144 }
145
146 override def compareTo(other: Version): Int = {
147 if (version == other.version) // String equals without requiring parse
148 0
149 else {
150 parse()
```

src/main/scala/akka/util/Version.scala, line 87 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: parseLastParts()
File: src/main/scala/akka/util/Version.scala:87
Taint Flags:

```
84 def parseLastParts(s: String): (Int, Int, String) = {
85 // for example 2, 2-SNAPSHOT or dynver 2+10-1234abcd
86 val (lastNumber, rest) = parseLastPart(s)
87 if (rest == "")
88 (lastNumber, Undefined, rest)
89 else {
90 val (dynverNumber, rest2) = parseDynverPart(rest)
```

Package: src.main.scala-2.13.akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 1246 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala-2.13/akka/util/ByteString.scala:1246
Taint Flags:

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala-2.13.akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 1246 (Code Correctness: Class Does Not Implement equals) — Low

```
1243  */
1244  def putLong(x: Long)(implicit byteOrder: ByteOrder): this.type = {
1245  fillArray(8) { (target, offset) =>
1246  if (byteOrder == ByteOrder.BIG_ENDIAN) {
1247  target(offset + 0) = (x >>> 56).toByte
1248  target(offset + 1) = (x >>> 48).toByte
1249  target(offset + 2) = (x >>> 40).toByte
```

src/main/scala-2.13/akka/util/ByteString.scala, line 1231 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala-2.13/akka/util/ByteString.scala:1231
Taint Flags:

```
1228  target(offset + 1) = (x >>> 16).toByte
1229  target(offset + 2) = (x >>> 8).toByte
1230  target(offset + 3) = (x >>> 0).toByte
1231  } else if (byteOrder == ByteOrder.LITTLE_ENDIAN) {
1232  target(offset + 0) = (x >>> 0).toByte
1233  target(offset + 1) = (x >>> 8).toByte
1234  target(offset + 2) = (x >>> 16).toByte
```

src/main/scala-2.13/akka/util/ByteString.scala, line 1255 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala-2.13/akka/util/ByteString.scala:1255
Taint Flags:

```
1252  target(offset + 5) = (x >>> 16).toByte
1253  target(offset + 6) = (x >>> 8).toByte
1254  target(offset + 7) = (x >>> 0).toByte
```

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala-2.13.akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 1255 (Code Correctness: Class Does Not Implement equals) — Low

```
1255  } else if (byteOrder == ByteOrder.LITTLE_ENDIAN) {
1256  target(offset + 0) = (x >>> 0).toByte
1257  target(offset + 1) = (x >>> 8).toByte
1258  target(offset + 2) = (x >>> 16).toByte
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 497 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:497
Taint Flags:

```
494  def indexOf(elem: Byte, from: Int): Int = indexWhere(_ == elem, from)
495
496  override def indexOf[B >: Byte](elem: B): Int = indexOf(elem, 0)
497  override def indexOf[B >: Byte](elem: B, from: Int): Int = indexWhere(_ == elem, from)
498
499  def toByteString: ByteString
500
```

src/main/scala-2.13/akka/util/ByteString.scala, line 1274 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala-2.13/akka/util/ByteString.scala:1274
Taint Flags:

```
1271  */
1272  def putLongPart(x: Long, n: Int)(implicit byteOrder: ByteOrder): this.type = {
1273  fillArray(n) { (target, offset) =>
1274  if (byteOrder == ByteOrder.BIG_ENDIAN) {
1275  val start = n * 8 - 8
1276  (0 until n).foreach { i =>
```

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala-2.13.akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 1274 (Code Correctness: Class Does Not Implement equals) — Low

```
1277  target(offset + i) = (x >>> start - 8 * i).toByte
```

src/main/scala-2.13/akka/util/ByteString.scala, line 1279 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala-2.13/akka/util/ByteString.scala:1279
Taint Flags:

```
1276  (0 until n).foreach { i =>
1277  target(offset + i) = (x >>> start - 8 * i).toByte
1278  }
1279  } else if (byteOrder == ByteOrder.LITTLE_ENDIAN) {
1280  (0 until n).foreach { i =>
1281  target(offset + i) = (x >>> 8 * i).toByte
1282  }
```

src/main/scala-2.13/akka/util/ByteString.scala, line 1226 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala-2.13/akka/util/ByteString.scala:1226
Taint Flags:

```
1223  */
1224  def putInt(x: Int)(implicit byteOrder: ByteOrder): this.type = {
1225  fillArray(4) { (target, offset) =>
1226  if (byteOrder == ByteOrder.BIG_ENDIAN) {
1227  target(offset + 0) = (x >>> 24).toByte
1228  target(offset + 1) = (x >>> 16).toByte
1229  target(offset + 2) = (x >>> 8).toByte
```

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 740 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:740
Taint Flags:

```
737 val deadline = Deadline.now + timeout
738 val timeoutFut = try {
739 after(timeout, system.scheduler) {
740 if (phaseName == CoordinatedShutdown.PhaseActorSystemTerminate && deadline.hasTimeLeft()) {
741 // too early, i.e. triggered by system termination
742 result
743 } else if (result.isCompleted)
```

src/main/scala/akka/actor/Address.scala, line 128 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/actor/Address.scala:128
Taint Flags:

```
125 */
126 implicit val addressOrdering: Ordering[Address] = Ordering.fromLessThan[Address] { (a, b) =>
127 if (a eq b) false
128 else if (a.protocol != b.protocol) a.system.compareTo(b.protocol) < 0
129 else if (a.system != b.system) a.system.compareTo(b.system) < 0
130 else if (a.host != b.host) a.host.getOrElse("").compareTo(b.host.getOrElse("")) < 0
131 else if (a.port != b.port) a.port.getOrElse(0) < b.port.getOrElse(0)
```

src/main/scala/akka/actor/Address.scala, line 129 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/Address.scala, line 129 (Code Correctness: Class Does Not Implement equals) — Low

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/actor/Address.scala:129
Taint Flags:

```
126 implicit val addressOrdering: Ordering[Address] = Ordering.fromLessThan[Address] { (a, b) =>
127 if (a eq b) false
128 else if (a.protocol != b.protocol) a.system.compareTo(b.protocol) < 0
129 else if (a.system != b.system) a.system.compareTo(b.system) < 0
130 else if (a.host != b.host) a.host.getOrElse("").compareTo(b.host.getOrElse("")) < 0
131 else if (a.port != b.port) a.port.getOrElse(0) < b.port.getOrElse(0)
132 else false
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 662 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/actor/ActorRefProvider.scala:662
Taint Flags:

```
659 case NoRouter =>
660 if (settings.DebugRouterMisconfiguration) {
661 deployer.lookup(path).foreach { d =>
662 if (d.routerConfig != NoRouter)
663 log.warning(
664 "Configuration says that [{}] should be a router, but code disagrees. Remove the config or add a routerConfig to its Props.",
665 path)
```

src/main/scala/akka/actor/ActorSystem.scala, line 355 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 355 (Code Correctness: Class Does Not Implement equals) — Low

File: src/main/scala/akka/actor/ActorSystem.scala:355
Taint Flags:

```
352 val confKey = "akka.use-slf4j"
353 if (config.hasPath(confKey) && config.getBoolean(confKey) && dynamicAccess.classIsOnClasspath(
354 slf4jLoggerClassName)) {
355 val newLoggers = slf4jLoggerClassName +: configuredLoggers.filterNot(_ == classOf[DefaultLogger].getName)
356 val newLoggersConfStr = s"$loggersConfKey = [${newLoggers.mkString("\"", "\", \"", "\"")}]"
357 val newConfStr =
358 if (loggingFilterAlreadyConfigured) newLoggersConfStr
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 768 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:768
Taint Flags:

```
765
766 val remainingPhases = fromPhase match {
767 case None => orderedPhases // all
768 case Some(p) => orderedPhases.dropWhile(_ != p)
769 }
770 val done = loop(remainingPhases)
771 runPromise.completeWith(done)
```

Package: src.main.scala.akka.event

src/main/scala/akka/event/Logging.scala, line 124 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/event/Logging.scala:124
Taint Flags:

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala.akka.event

src/main/scala/akka/event/Logging.scala, line 124 (Code Correctness: Class Does Not Implement equals) — Low

```
121 val myloggers =
122 for {
123 loggerName <- defaultLoggers
124 if loggerName != StandardOutLogger.getClass.getName
125 } yield {
126 system.dynamicAccess
127 .getClassFor[Actor](loggerName)
```

Package: src.main.scala.akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 55 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:55
Taint Flags:

```
52
53 private lazy val defaultCachePolicy: CachePolicy =
54 Option(Security.getProperty(CachePolicyProp))
55 .filter(_ != "")
56 .orElse(Option(System.getProperty(CachePolicyPropFallback)))
57 .filter(_ != "")
58 .map(x => Try(x.toInt)) match {
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 57 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:57
Taint Flags:

```
54 Option(Security.getProperty(CachePolicyProp))
55 .filter(_ != "")
```

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala.akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 57 (Code Correctness: Class Does Not Implement equals) — Low

```
56  .orElse(Option(System.getProperty(CachePolicyPropFallback)))
57  .filter(_ != "")
58  .map(x => Try(x.toInt)) match {
59  case None => DefaultPositive
60  case Some(Success(n)) => parsePolicy(n)
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 68 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:68
Taint Flags:

```
65
66  private lazy val defaultNegativeCachePolicy: CachePolicy =
67  Option(Security.getProperty(NegativeCachePolicyProp))
68  .filter(_ != "")
69  .orElse(Option(System.getProperty(NegativeCachePolicyPropFallback)))
70  .filter(_ != "")
71  .map(x => Try(x.toInt)) match {
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 70 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:70
Taint Flags:

```
67  Option(Security.getProperty(NegativeCachePolicyProp))
68  .filter(_ != "")
69  .orElse(Option(System.getProperty(NegativeCachePolicyPropFallback)))
70  .filter(_ != "")
71  .map(x => Try(x.toInt)) match {
```

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala.akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 70 (Code Correctness: Class Does Not Implement equals) — Low

```
72 case None => Never
73 case Some(Success(n)) => parsePolicy(n)
```

Package: src.main.scala.akka.io.dns.internal

src/main/scala/akka/io/dns/internal/ResolvConfParser.scala, line 51 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/io/dns/internal/ResolvConfParser.scala:51
Taint Flags:

```
48 label match {
49 case `DomainLabel` =>
50 search = List(trimmedArgs)
51 case `SearchLabel` =>
52 search = trimmedArgs.split("\\s+").toList
53 case `OptionsLabel` =>
54 args.split("\\s+").foreach { option =>
```

src/main/scala/akka/io/dns/internal/ResolvConfParser.scala, line 49 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/io/dns/internal/ResolvConfParser.scala:49
Taint Flags:

```
46 val (label, args) = line.span(!_.isWhitespace)
47 def trimmedArgs = args.trim
48 label match {
49 case `DomainLabel` =>
50 search = List(trimmedArgs)
51 case `SearchLabel` =>
52 search = trimmedArgs.split("\\s+").toList
```

Code Correctness: Class Does Not Implement equals	Low

Package: src.main.scala.akka.io.dns.internal

src/main/scala/akka/io/dns/internal/ResolvConfParser.scala, line 49 (Code Correctness: Class Does Not Implement equals) **Low**

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 101 (Code Correctness: Class Does Not Implement equals) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala:101
Taint Flags:

```
98  if (resolved.records.nonEmpty) {
99  val minTtl = (positiveCachePolicy +: resolved.records.map(_.ttl)).min
100 cache.put((name, mode), resolved, minTtl)
101 } else if (negativeCachePolicy != Never) cache.put((name, mode), resolved, negativeCachePolicy)
102 log.debug(s"{} resolved {}", mode, resolved)
103 resolved
104 }
```

src/main/scala/akka/io/dns/internal/ResolvConfParser.scala, line 53 (Code Correctness: Class Does Not Implement equals) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/io/dns/internal/ResolvConfParser.scala:53
Taint Flags:

```
50  search = List(trimmedArgs)
51  case `SearchLabel` =>
52  search = trimmedArgs.split("\\s+").toList
53  case `OptionsLabel` =>
54  args.split("\\s+").foreach { option =>
55  // We're only interested in ndots
56  if (option.startsWith(NdotsOption)) {
```

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala.akka.pattern

src/main/scala/akka/pattern/GracefulStopSupport.scala, line 56 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/pattern/GracefulStopSupport.scala:56
Taint Flags:

```
53 internalTarget.sendSystemMessage(Watch(internalTarget, ref))
54 target.tell(stopMessage, Actor.noSender)
55 ref.result.future.transform({
56 case Terminated(t) if t.path == target.path => true
57 case _ => { internalTarget.sendSystemMessage(Unwatch(target, ref)); false }
58 }, t => { internalTarget.sendSystemMessage(Unwatch(target, ref)); t })(ExecutionContexts.parasitic)
59 }
```

Package: src.main.scala.akka.routing

src/main/scala/akka/routing/Router.scala, line 161 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/routing/Router.scala:161
Taint Flags:

```
158 /**
159 * Create a new instance without the specified routee.
160 */
161 def removeRoutee(routee: Routee): Router = copy(routees = routees.filterNot(_ == routee))
162
163 /**
164 * Create a new instance without the [[ActorRefRoutee]] for the specified
```

src/main/scala/akka/routing/RoutedActorCell.scala, line 77 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala.akka.routing

src/main/scala/akka/routing/RoutedActorCell.scala, line 77 (Code Correctness: Class Does Not Implement equals) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/routing/RoutedActorCell.scala:77
Taint Flags:

```
74 def removeRoutees(routees: immutable.Iterable[Routee], stopChild: Boolean): Unit = {
75 val r = _router
76 val newRoutees = routees.foldLeft(r.routees) { (xs, x) =>
77 unwatch(x); xs.filterNot(_ == x)
78 }
79 _router = r.withRoutees(newRoutees)
80 if (stopChild) routees.foreach(stopIfChild)
```

Package: src.main.scala.akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 512 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/serialization/Serialization.scala:512
Taint Flags:

```
509 case (acc, (_, ser)) =>
510 val id = ser.identifier
511 acc.get(id) match {
512 case Some(existing) if existing != ser =>
513 throw new IllegalArgumentException(
514 s"Serializer identifier [$id] of [${ser.getClass.getName}] " +
515 s"is not unique. It is also used by [${acc(id).getClass.getName}].")
```

src/main/scala/akka/serialization/Serialization.scala, line 220 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Class Does Not Implement equals — Low

Package: src.main.scala.akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 220 (Code Correctness: Class Does Not Implement equals) — Low

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/serialization/Serialization.scala:220
Taint Flags:

```
217 serializer match {
218 case s2: SerializerWithStringManifest => s2.fromBinary(bytes, manifest)
219 case s1 =>
220 if (manifest == "")
221 s1.fromBinary(bytes, None)
222 else {
223 val cache = manifestCache.get
```

src/main/scala/akka/serialization/Serialization.scala, line 437 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals
Enclosing Method: apply()
File: src/main/scala/akka/serialization/Serialization.scala:437
Taint Flags:

```
434 private[akka] val bindings: immutable.Seq[ClassSerializer] = {
435 val fromConfig = for {
436 (className: String, alias: String) <- settings.SerializationBindings
437 if alias != "none" && checkGoogleProtobuf(className) && checkAkkaProtobuf(className)
438 } yield (system.dynamicAccess.getClassFor[Any](className).get, serializers(alias))
439
440 val fromSettings = serializerDetails.flatMap { detail =>
```

Package: src.main.scala.akka.util

src/main/scala/akka/util/LineNumbers.scala, line 304 (Code Correctness: Class Does Not Implement equals) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: equals

Code Correctness: Class Does Not Implement equals	Low

Package: src.main.scala.akka.util

src/main/scala/akka/util/LineNumbers.scala, line 304 (Code Correctness: Class Does Not Implement equals)	Low

Enclosing Method: apply()
File: src/main/scala/akka/util/LineNumbers.scala:304
Taint Flags:

```
301 for (_ <- 1 to d.readUnsignedShort()) yield {
302 val tag = d.readUnsignedShort()
303 val length = d.readInt()
304 if (tag != codeTag || (filter.isDefined && c(name) != filter.get)) {
305 skip(d, length)
306 None
307 } else {
```

Code Correctness: Constructor Invokes Overridable Function (316 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.

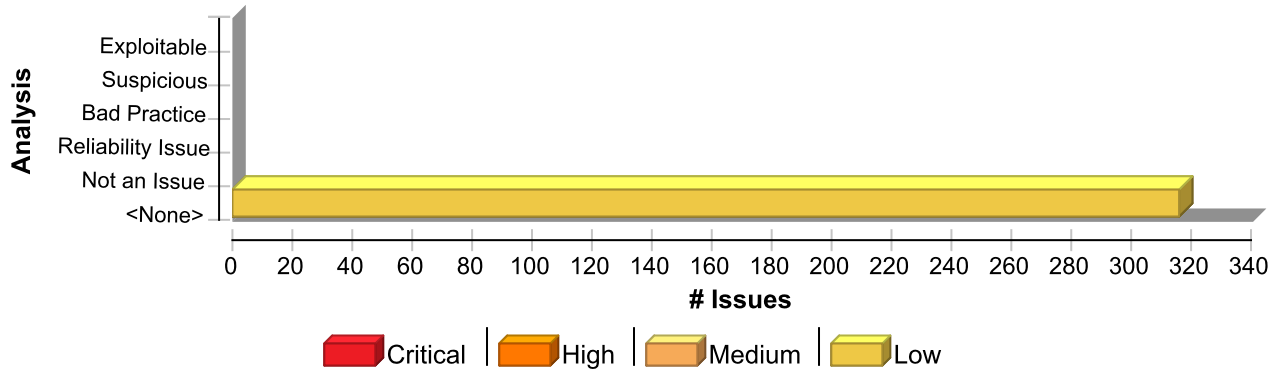
Recommendation

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	316	0	0	316
Total	**316**	**0**	**0**	**316**

Code Correctness: Constructor Invokes Overridable Function	**Low**
Package: akka.actor	
src/main/scala/akka/actor/Deployer.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function)	**Low**
Issue Details	

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Deployer.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:82
Taint Flags:

```
79 /**
80 * Java API to create a Deploy with the given RouterConfig
81 */
82 def this(routing: RouterConfig) = this("", ConfigFactory.empty, routing)
83
84 /**
85 * Java API to create a Deploy with the given RouterConfig with Scope
```

src/main/scala/akka/actor/FaultHandling.scala, line 618 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: OneForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:618
Taint Flags:

```
615 * Java API
616 */
617 def this(maxNrOfRetries: Int, withinTimeRange: java.time.Duration, trapExit: JIterable[Class[_ <: Throwable]]) =
618 this(maxNrOfRetries, withinTimeRange.asScala)(SupervisorStrategy.makeDecider(trapExit))
619
620 /**
621 * Java API: compatible with lambda expressions
```

src/main/scala/akka/actor/FaultHandling.scala, line 512 (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.actor

src/main/scala/akka/actor/FaultHandling.scala, line 512 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: AllForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:512
Taint Flags:

```
509 * Java API: compatible with lambda expressions
510 */
511 def this(maxNrOfRetries: Int, withinTimeRange: Duration, decider: SupervisorStrategy.Decider) =
512 this(maxNrOfRetries = maxNrOfRetries, withinTimeRange = withinTimeRange)(decider)
513
514 /**
515 * Java API: compatible with lambda expressions
```

src/main/scala/akka/actor/FaultHandling.scala, line 530 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: AllForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:530
Taint Flags:

```
527 * Java API: compatible with lambda expressions
528 */
529 def this(decider: SupervisorStrategy.Decider) =
530 this()(decider)
531
532 /*
533 * this is a performance optimization to avoid re-allocating the pairs upon
```

src/main/scala/akka/actor/Scheduler.scala, line 542 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: Cancellable$$anon$8
Enclosing Method: Cancellable()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Scheduler.scala, line 542 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/Scheduler.scala:542
Taint Flags:

```
539 }
540
541 object Cancellable {
542  val alreadyCancelled: Cancellable = new Cancellable {
543  def cancel(): Boolean = false
544  def isCancelled: Boolean = true
545  }
```

src/main/scala/akka/actor/ActorSystem.scala, line 938 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: liftedTree1
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:938
Taint Flags:

```
935
936  val scheduler: Scheduler = createScheduler()
937
938  val provider: ActorRefProvider = try {
939  val arguments = Vector(
940  classOf[String] -> name,
941  classOf[Settings] -> settings,
```

src/main/scala/akka/actor/ActorSystem.scala, line 970 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: provider
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:970
Taint Flags:

```
967
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 970 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
968 val dispatcher: ExecutionContextExecutor = dispatchers.defaultGlobalDispatcher
969
970 private[this] final val terminationCallbacks = new TerminationCallbacks(provider.terminationFuture)(dispatcher)
971
972 override def whenTerminated: Future[Terminated] = terminationCallbacks.terminationFuture
973 override def getWhenTerminated: CompletionStage[Terminated] = FutureConverters.toJava(whenTerminated)
```

src/main/scala/akka/actor/FaultHandling.scala, line 624 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: OneForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:624
Taint Flags:

```
621 * Java API: compatible with lambda expressions
622 */
623 def this(maxNrOfRetries: Int, withinTimeRange: Duration, decider: SupervisorStrategy.Decider) =
624 this(maxNrOfRetries = maxNrOfRetries, withinTimeRange = withinTimeRange)(decider)
625
626 /**
627 * Java API: compatible with lambda expressions
```

src/main/scala/akka/actor/TypedActor.scala, line 608 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:608
Taint Flags:

```
605 * appended in the sequence of interfaces.
606 */
607 def this(interface: Class[_ >: T], implementation: Creator[T]) =
608 this(interfaces = TypedProps.extractInterfaces(interface), creator = implementation.create _)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/TypedActor.scala, line 608 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
609
610 /**
611 * Java API: Uses the supplied class as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/Deployer.scala, line 21 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply$default$5
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:21
Taint Flags:

```
18 object Deploy {
19 final val NoDispatcherGiven = ""
20 final val NoMailboxGiven = ""
21 val local = Deploy(scope = LocalScope)
22
23 /**
24 * INTERNAL API
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 189 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: WheelSize
Enclosing Method: LightArrayRevolverScheduler()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:189
Taint Flags:

```
186
187 private val start = clock()
188 private val tickNanos = TickDuration.toNanos
189 private val wheelMask = WheelSize - 1
190 private val queue = new TaskQueue
191
192 private def schedule(ec: ExecutionContext, r: Runnable, delay: FiniteDuration): TimerTask =
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Props.scala, line 47 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: defaultDeploy
Enclosing Method: Props()
File: src/main/scala/akka/actor/Props.scala:47
Taint Flags:

```
44 /**
45 * The default Props instance, uses the settings from the Props object starting with default*.
46 */
47 final val default = Props(defaultDeploy, classOf[CreatorFunctionConsumer], List(defaultCreator))
48
49 /**
50 * INTERNAL API
```

src/main/scala/akka/actor/TypedActor.scala, line 617 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:617
Taint Flags:

```
614 * appended in the sequence of interfaces.
615 */
616 def this(interface: Class[_ >: T], implementation: Class[T]) =
617 this(interfaces = TypedProps.extractInterfaces(interface), creator = instantiator(implementation))
618
619 /**
620 * Returns a new TypedProps with the specified dispatcher set.
```

src/main/scala/akka/actor/ActorRef.scala, line 361 (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.actor

src/main/scala/akka/actor/ActorRef.scala, line 361 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: actorCell
Enclosing Method: LocalActorRef()
File: src/main/scala/akka/actor/ActorRef.scala:361
Taint Flags:

```
358 * object from another thread as soon as we run init.
359 */
360 private val actorCell: ActorCell = newActorCell(_system, this, _props, _dispatcher, _supervisor)
361 actorCell.init(sendSupervise = true, _mailboxType)
362
363 protected def newActorCell(
364 system: ActorSystemImpl,
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 404 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: rootPath
Enclosing Method: LocalActorRefProvider()
File: src/main/scala/akka/actor/ActorRefProvider.scala:404
Taint Flags:

```
401 private[akka] val logDeser: MarkerLoggingAdapter =
402 Logging.withMarker(eventStream, getClass.getName + ".Deserialization")
403
404 override val deadLetters: InternalActorRef =
405 _deadLetters
406 .getOrElse((p: ActorPath) => new DeadLetterActorRef(this, p, eventStream))
407 .apply(rootPath / "deadLetters")
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 420 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: rootPath
Enclosing Method: LocalActorRefProvider()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRefProvider.scala, line 420 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/ActorRefProvider.scala:420
Taint Flags:

```
417 */
418 private val tempNumber = new AtomicLong
419
420 private val tempNode = rootPath / "temp"
421
422 override def tempPath(): ActorPath = tempPath("")
423
```

src/main/scala/akka/actor/ActorSystem.scala, line 968 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: dispatchers
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:968
Taint Flags:

```
965 defaultExecutionContext),
966 log)
967
968 val dispatcher: ExecutionContextExecutor = dispatchers.defaultGlobalDispatcher
969
970 private[this] final val terminationCallbacks = new TerminationCallbacks(provider.terminationFuture)(dispatcher)
971
```

src/main/scala/akka/actor/Deployer.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:82
Taint Flags:

```
79 /**
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Deployer.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
80  * Java API to create a Deploy with the given RouterConfig
81  */
82  def this(routing: RouterConfig) = this("", ConfigFactory.empty, routing)
83
84  /**
85  * Java API to create a Deploy with the given RouterConfig with Scope
```

src/main/scala/akka/actor/Props.scala, line 47 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: defaultCreator
Enclosing Method: Props()
File: src/main/scala/akka/actor/Props.scala:47
Taint Flags:

```
44  /**
45  * The default Props instance, uses the settings from the Props object starting with default*.
46  */
47  final val default = Props(defaultDeploy, classOf[CreatorFunctionConsumer], List(defaultCreator))
48
49  /**
50  * INTERNAL API
```

src/main/scala/akka/actor/ActorSystem.scala, line 819 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: createDynamicAccess
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:819
Taint Flags:

```
816
817  @volatile private var logDeadLetterListener: Option[ActorRef] = None
818
819  private val _dynamicAccess: DynamicAccess = createDynamicAccess()
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 819 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
820
821 final val settings: Settings = {
822 val config = Settings.amendSlf4jConfig(
```

src/main/scala/akka/actor/TypedActor.scala, line 608 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:608
Taint Flags:

```
605 * appended in the sequence of interfaces.
606 */
607 def this(interface: Class[_ >: T], implementation: Creator[T]) =
608 this(interfaces = TypedProps.extractInterfaces(interface), creator = implementation.create _)
609
610 /**
611 * Java API: Uses the supplied class as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/FaultHandling.scala, line 494 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: AllForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:494
Taint Flags:

```
491 * Java API
492 */
493 def this(maxNrOfRetries: Int, withinTimeRange: java.time.Duration, decider: SupervisorStrategy.JDecider) =
494 this(maxNrOfRetries, withinTimeRange.asScala)(SupervisorStrategy.makeDecider(decider))
495
496 /**
497 * Java API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 524 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: AllForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:524
Taint Flags:

```
521 * Java API: compatible with lambda expressions
522 */
523 def this(loggingEnabled: Boolean, decider: SupervisorStrategy.Decider) =
524 this(loggingEnabled = loggingEnabled)(decider)
525
526 /**
527 * Java API: compatible with lambda expressions
```

src/main/scala/akka/actor/FaultHandling.scala, line 518 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: AllForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:518
Taint Flags:

```
515 * Java API: compatible with lambda expressions
516 */
517 def this(maxNrOfRetries: Int, withinTimeRange: java.time.Duration, decider: SupervisorStrategy.Decider) =
518 this(maxNrOfRetries = maxNrOfRetries, withinTimeRange = withinTimeRange.asScala)(decider)
519
520 /**
521 * Java API: compatible with lambda expressions
```

src/main/scala/akka/actor/ActorSystem.scala, line 970 (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.actor

src/main/scala/akka/actor/ActorSystem.scala, line 970 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: dispatcher
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:970
Taint Flags:

```
967
968 val dispatcher: ExecutionContextExecutor = dispatchers.defaultGlobalDispatcher
969
970 private[this] final val terminationCallbacks = new TerminationCallbacks(provider.terminationFuture)(dispatcher)
971
972 override def whenTerminated: Future[Terminated] = terminationCallbacks.terminationFuture
973 override def getWhenTerminated: CompletionStage[Terminated] = FutureConverters.toJava(whenTerminated)
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 335 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: timerThread
Enclosing Method: LightArrayRevolverScheduler()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:335
Taint Flags:

```
332 }
333 })
334
335 timerThread.start()
336 }
337
338 object LightArrayRevolverScheduler {
```

src/main/scala/akka/actor/ActorSystem.scala, line 881 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ActorSystemImpl()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 881 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/ActorSystem.scala:881
Taint Flags:

```
878 }
879
880 final val threadFactory: MonitorableThreadFactory =
881 MonitorableThreadFactory(name, settings.Daemonicity, Option(classLoader), uncaughtExceptionHandler)
882
883 /**
884 * This is an extension point: by overriding this method, subclasses can
```

src/main/scala/akka/actor/ActorSystem.scala, line 924 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:924
Taint Flags:

```
921 import settings._
922
923 // this provides basic logging (to stdout) until .start() is called below
924 val eventStream = new EventStream(this, DebugEventStream)
925 eventStream.startStdoutLogger(settings)
926
927 val logFilter: LoggingFilter = {
```

src/main/scala/akka/actor/ActorSystem.scala, line 925 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:925
Taint Flags:

```
922
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 925 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
923 // this provides basic logging (to stdout) until .start() is called below
924 val eventStream = new EventStream(this, DebugEventStream)
925 eventStream.startStdoutLogger(settings)
926
927 val logFilter: LoggingFilter = {
928 val arguments = Vector(classOf[Settings] -> settings, classOf[EventStream] -> eventStream)
```

src/main/scala/akka/actor/ActorSystem.scala, line 927 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:927
Taint Flags:

```
924 val eventStream = new EventStream(this, DebugEventStream)
925 eventStream.startStdoutLogger(settings)
926
927 val logFilter: LoggingFilter = {
928 val arguments = Vector(classOf[Settings] -> settings, classOf[EventStream] -> eventStream)
929 dynamicAccess.createInstanceFor[LoggingFilter](LoggingFilter, arguments).get
930 }
```

src/main/scala/akka/actor/ActorSystem.scala, line 928 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:928
Taint Flags:

```
925 eventStream.startStdoutLogger(settings)
926
927 val logFilter: LoggingFilter = {
928 val arguments = Vector(classOf[Settings] -> settings, classOf[EventStream] -> eventStream)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 928 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
929 dynamicAccess.createInstanceFor[LoggingFilter](LoggingFilter, arguments).get
930 }
931
```

src/main/scala/akka/actor/ActorSystem.scala, line 954 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:954
Taint Flags:

```
951
952 def deadLetters: ActorRef = provider.deadLetters
953
954 val mailboxes: Mailboxes = new Mailboxes(settings, eventStream, dynamicAccess, deadLetters)
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
```

src/main/scala/akka/actor/ActorSystem.scala, line 956 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:956
Taint Flags:

```
953
954 val mailboxes: Mailboxes = new Mailboxes(settings, eventStream, dynamicAccess, deadLetters)
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
958 DefaultDispatcherPrerequisites(
959 threadFactory,
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 958 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:958
Taint Flags:

```
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
958 DefaultDispatcherPrerequisites(
959 threadFactory,
960 eventStream,
961 scheduler,
```

src/main/scala/akka/actor/Deployer.scala, line 21 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply$default$6
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:21
Taint Flags:

```
18 object Deploy {
19 final val NoDispatcherGiven = ""
20 final val NoMailboxGiven = ""
21 val local = Deploy(scope = LocalScope)
22
23 /**
24 * INTERNAL API
```

src/main/scala/akka/actor/ActorRef.scala, line 537 (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.actor

src/main/scala/akka/actor/ActorRef.scala, line 537 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: path
Enclosing Method: IgnoreActorRef()
File: src/main/scala/akka/actor/ActorRef.scala:537
Taint Flags:

```
534 val path: ActorPath =
535 RootActorPath(Address("akka", IgnoreActorRef.fakeSystemName)) / "ignore"
536
537 private val pathString = path.toString
538
539 /**
540 * Check if the passed `otherPath` is the same as IgnoreActorRef.path
```

src/main/scala/akka/actor/TypedActor.scala, line 599 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:599
Taint Flags:

```
596 * appended in the sequence of interfaces.
597 */
598 def this(implementation: Class[T]) =
599 this(interfaces = TypedProps.extractInterfaces(implementation), creator = instantiator(implementation))
600
601 /**
602 * Java API: Uses the supplied Creator as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/FaultHandling.scala, line 633 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: OneForOneStrategy()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 633 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/FaultHandling.scala:633
Taint Flags:

```
630 this(maxNrOfRetries = maxNrOfRetries, withinTimeRange = withinTimeRange.asScala)(decider)
631
632 def this(loggingEnabled: Boolean, decider: SupervisorStrategy.Decider) =
633 this(loggingEnabled = loggingEnabled)(decider)
634
635 /**
636 * Java API: Restart an infinite number of times. Compatible with lambda expressions.
```

src/main/scala/akka/actor/ActorSystem.scala, line 927 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: dynamicAccess
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:927
Taint Flags:

```
924 val eventStream = new EventStream(this, DebugEventStream)
925 eventStream.startStdoutLogger(settings)
926
927 val logFilter: LoggingFilter = {
928 val arguments = Vector(classOf[Settings] -> settings, classOf[EventStream] -> eventStream)
929 dynamicAccess.createInstanceFor[LoggingFilter](LoggingFilter, arguments).get
930 }
```

src/main/scala/akka/actor/ActorSystem.scala, line 954 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: dynamicAccess
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:954
Taint Flags:

```
951
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 954 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
952 def deadLetters: ActorRef = provider.deadLetters
953
954 val mailboxes: Mailboxes = new Mailboxes(settings, eventStream, dynamicAccess, deadLetters)
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
```

src/main/scala/akka/actor/ActorSystem.scala, line 958 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: dynamicAccess
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:958
Taint Flags:

```
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
958 DefaultDispatcherPrerequisites(
959 threadFactory,
960 eventStream,
961 scheduler,
```

src/main/scala/akka/actor/Deployer.scala, line 87 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:87
Taint Flags:

```
84 /**
85 * Java API to create a Deploy with the given RouterConfig with Scope
86 */
87 def this(routing: RouterConfig, scope: Scope) = this("", ConfigFactory.empty, routing, scope)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Deployer.scala, line 87 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
88
89 /**
90 * Java API to create a Deploy with the given Scope
```

src/main/scala/akka/actor/ActorRef.scala, line 535 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: fakeSystemName
Enclosing Method: IgnoreActorRef()
File: src/main/scala/akka/actor/ActorRef.scala:535
Taint Flags:

```
532 private val fakeSystemName = "local"
533
534 val path: ActorPath =
535 RootActorPath(Address("akka", IgnoreActorRef.fakeSystemName)) / "ignore"
536
537 private val pathString = path.toString
538
```

src/main/scala/akka/actor/Deployer.scala, line 21 (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: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:21
Taint Flags:

```
18 object Deploy {
19 final val NoDispatcherGiven = ""
20 final val NoMailboxGiven = ""
21 val local = Deploy(scope = LocalScope)
22
23 /**
24 * INTERNAL API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Deployer.scala, line 87 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:87
Taint Flags:

```
84 /**
85 * Java API to create a Deploy with the given RouterConfig with Scope
86 */
87 def this(routing: RouterConfig, scope: Scope) = this("", ConfigFactory.empty, routing, scope)
88
89 /**
90 * Java API to create a Deploy with the given Scope
```

src/main/scala/akka/actor/ActorPath.scala, line 371 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: toStringLength
Enclosing Method: ChildActorPath()
File: src/main/scala/akka/actor/ActorPath.scala:371
Taint Flags:

```
368
369 private val toStringOffset: Int = parent match {
370 case r: RootActorPath => r.address.toString.length + r.name.length
371 case c: ChildActorPath => c.toStringLength + 1
372 }
373
374 override def toStringWithAddress(addr: Address): String = {
```

src/main/scala/akka/actor/Deployer.scala, line 92 (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.actor

src/main/scala/akka/actor/Deployer.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:92
Taint Flags:

```
89 /**
90 * Java API to create a Deploy with the given Scope
91 */
92 def this(scope: Scope) = this("", ConfigFactory.empty, NoRouter, scope)
93
94 /**
95 * Do a merge between this and the other Deploy, where values from “this” take
```

src/main/scala/akka/actor/Props.scala, line 144 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: producer
Enclosing Method: Props()
File: src/main/scala/akka/actor/Props.scala:144
Taint Flags:

```
141 }
142
143 // validate producer constructor signature; throws IllegalArgumentException if invalid
144 producer
145
146 /**
147 * Convenience method for extracting the dispatcher information from the
```

src/main/scala/akka/actor/FaultHandling.scala, line 488 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: AllForOneStrategy()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 488 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/FaultHandling.scala:488
Taint Flags:

```
485 * Java API
486 */
487 def this(maxNrOfRetries: Int, withinTimeRange: Duration, decider: SupervisorStrategy.JDecider) =
488 this(maxNrOfRetries, withinTimeRange)(SupervisorStrategy.makeDecider(decider))
489
490 /**
491 * Java API
```

src/main/scala/akka/actor/FaultHandling.scala, line 530 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: AllForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:530
Taint Flags:

```
527 * Java API: compatible with lambda expressions
528 */
529 def this(decider: SupervisorStrategy.Decider) =
530 this()(decider)
531
532 /*
533 * this is a performance optimization to avoid re-allocating the pairs upon
```

src/main/scala/akka/actor/Deployer.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$7
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:92
Taint Flags:

```
89 /**
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Deployer.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
90  * Java API to create a Deploy with the given Scope
91  */
92  def this(scope: Scope) = this("", ConfigFactory.empty, NoRouter, scope)
93
94  /**
95  * Do a merge between this and the other Deploy, where values from “this” take
```

src/main/scala/akka/actor/TypedActor.scala, line 608 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:608
Taint Flags:

```
605  * appended in the sequence of interfaces.
606  */
607  def this(interface: Class[_ >: T], implementation: Creator[T]) =
608  this(interfaces = TypedProps.extractInterfaces(interface), creator = implementation.create _)
609
610  /**
611  * Java API: Uses the supplied class as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/FaultHandling.scala, line 606 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: OneForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:606
Taint Flags:

```
603  * Java API
604  */
605  def this(maxNrOfRetries: Int, withinTimeRange: java.time.Duration, decider: SupervisorStrategy.JDecider) =
606  this(maxNrOfRetries, withinTimeRange.asScala)(SupervisorStrategy.makeDecider(decider))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 606 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
607
608 /**
609 * Java API
```

src/main/scala/akka/actor/ActorSystem.scala, line 881 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: uncaughtExceptionHandler
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:881
Taint Flags:

```
878 }
879
880 final val threadFactory: MonitorableThreadFactory =
881 MonitorableThreadFactory(name, settings.Daemonicity, Option(classLoader), uncaughtExceptionHandler)
882
883 /**
884 * This is an extension point: by overriding this method, subclasses can
```

src/main/scala/akka/actor/ActorSystem.scala, line 925 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: eventStream
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:925
Taint Flags:

```
922
923 // this provides basic logging (to stdout) until .start() is called below
924 val eventStream = new EventStream(this, DebugEventStream)
925 eventStream.startStdoutLogger(settings)
926
927 val logFilter: LoggingFilter = {
928 val arguments = Vector(classOf[Settings] -> settings, classOf[EventStream] -> eventStream)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 928 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: eventStream
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:928
Taint Flags:

```
925 eventStream.startStdoutLogger(settings)
926
927 val logFilter: LoggingFilter = {
928 val arguments = Vector(classOf[Settings] -> settings, classOf[EventStream] -> eventStream)
929 dynamicAccess.createInstanceFor[LoggingFilter](LoggingFilter, arguments).get
930 }
931
```

src/main/scala/akka/actor/ActorSystem.scala, line 933 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: eventStream
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:933
Taint Flags:

```
930 }
931
932 private[this] val markerLogging =
933 new MarkerLoggingAdapter(eventStream, getClass.getName + "(" + name + ")", this.getClass, logFilter)
934 val log: LoggingAdapter = markerLogging
935
936 val scheduler: Scheduler = createScheduler()
```

src/main/scala/akka/actor/ActorSystem.scala, line 954 (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.actor

src/main/scala/akka/actor/ActorSystem.scala, line 954 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: eventStream
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:954
Taint Flags:

```
951
952 def deadLetters: ActorRef = provider.deadLetters
953
954 val mailboxes: Mailboxes = new Mailboxes(settings, eventStream, dynamicAccess, deadLetters)
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
```

src/main/scala/akka/actor/ActorSystem.scala, line 958 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: eventStream
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:958
Taint Flags:

```
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
958 DefaultDispatcherPrerequisites(
959 threadFactory,
960 eventStream,
961 scheduler,
```

src/main/scala/akka/actor/Actor.scala, line 189 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akka$actor$ActorInitializationException$$enrichedMessage
Enclosing Method: ActorInitializationException()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Actor.scala, line 189 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/Actor.scala:189
Taint Flags:

```
186 */
187 @SerialVersionUID(1L)
188 class ActorInitializationException protected (actor: ActorRef, message: String, cause: Throwable)
189  extends AkkaException(ActorInitializationException.enrichedMessage(actor, message), cause) {
190  def getActor: ActorRef = actor
191 }
192 object ActorInitializationException {
```

src/main/scala/akka/actor/ActorSystem.scala, line 956 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: log
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:956
Taint Flags:

```
953
954 val mailboxes: Mailboxes = new Mailboxes(settings, eventStream, dynamicAccess, deadLetters)
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
958 DefaultDispatcherPrerequisites(
959 threadFactory,
```

src/main/scala/akka/actor/ActorRef.scala, line 521 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: path
Enclosing Method: IgnoreActorRef()
File: src/main/scala/akka/actor/ActorRef.scala:521
Taint Flags:

```
518 */
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 521 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
519 @InternalApi private[akka] final class IgnoreActorRef(override val provider: ActorRefProvider) extends MinimalActorRef {
520
521 override val path: ActorPath = IgnoreActorRef.path
522
523 @throws(classOf[java.io.ObjectStreamException])
524 override protected def writeReplace(): AnyRef = SerializedIgnore
```

src/main/scala/akka/actor/Deployer.scala, line 222 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: Deployer()
File: src/main/scala/akka/actor/Deployer.scala:222
Taint Flags:

```
219 private val resizerEnabled: Config = ConfigFactory.parseString("resizer.enabled=on")
220 private val deployments = new AtomicReference(WildcardIndex[Deploy]())
221 private val config = settings.config.getConfig("akka.actor.deployment")
222 protected val default = config.getConfig("default")
223 val routerTypeMapping: Map[String, String] =
224 settings.config
225 .getConfig("akka.actor.router.type-mapping")
```

src/main/scala/akka/actor/Deployer.scala, line 240 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: Deployer()
File: src/main/scala/akka/actor/Deployer.scala:240
Taint Flags:

```
237 case (key, value: ConfigObject) => parseConfig(key, value.toConfig)
238 case _ => None
239 }
240 .foreach(deploy)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Deployer.scala, line 240 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
241
242  def lookup(path: ActorPath): Option[Deploy] = lookup(path.elements.drop(1))
243
```

src/main/scala/akka/actor/Deployer.scala, line 21 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:21
Taint Flags:

```
18 object Deploy {
19  final val NoDispatcherGiven = ""
20  final val NoMailboxGiven = ""
21  val local = Deploy(scope = LocalScope)
22
23  /**
24  * INTERNAL API
```

src/main/scala/akka/actor/ActorSystem.scala, line 958 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: threadFactory
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:958
Taint Flags:

```
955
956  val dispatchers: Dispatchers = new Dispatchers(
957  settings,
958  DefaultDispatcherPrerequisites(
959  threadFactory,
960  eventStream,
961  scheduler,
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 612 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: OneForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:612
Taint Flags:

```
609 * Java API
610 */
611 def this(maxNrOfRetries: Int, withinTimeRange: Duration, trapExit: JIterable[Class[_ <: Throwable]]) =
612 this(maxNrOfRetries, withinTimeRange)(SupervisorStrategy.makeDecider(trapExit))
613
614 /**
615 * Java API
```

src/main/scala/akka/actor/Scheduler.scala, line 550 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: Cancellable$$anon$9
Enclosing Method: Cancellable()
File: src/main/scala/akka/actor/Scheduler.scala:550
Taint Flags:

```
547 /**
548 * INTERNAL API
549 */
550 @InternalApi private[akka] val initialNotCancelled: Cancellable = new Cancellable {
551 def cancel(): Boolean = false
552 def isCancelled: Boolean = false
553 }
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 388 (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.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 388 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: topologicalSort
Enclosing Method: CoordinatedShutdown()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:388
Taint Flags:

```
385 private val knownPhases = phases.keySet ++ phases.values.flatMap(_.dependsOn)
386
387 /** INTERNAL API */
388 private[akka] val orderedPhases = CoordinatedShutdown.topologicalSort(phases)
389
390 private trait PhaseDefinition {
391 def size: Int
```

src/main/scala/akka/actor/FaultHandling.scala, line 524 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$1
Enclosing Method: AllForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:524
Taint Flags:

```
521 * Java API: compatible with lambda expressions
522 */
523 def this(loggingEnabled: Boolean, decider: SupervisorStrategy.Decider) =
524 this(loggingEnabled = loggingEnabled)(decider)
525
526 /**
527 * Java API: compatible with lambda expressions
```

src/main/scala/akka/actor/ActorSystem.scala, line 936 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: createScheduler
Enclosing Method: ActorSystemImpl()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 936 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/ActorSystem.scala:936
Taint Flags:

```
933 new MarkerLoggingAdapter(eventStream, getClass.getName + "(" + name + ")", this.getClass, logFilter)
934 val log: LoggingAdapter = markerLogging
935
936 val scheduler: Scheduler = createScheduler()
937
938 val provider: ActorRefProvider = try {
939 val arguments = Vector(
```

src/main/scala/akka/actor/TypedActor.scala, line 599 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:599
Taint Flags:

```
596 * appended in the sequence of interfaces.
597 */
598 def this(implementation: Class[T]) =
599 this(interfaces = TypedProps.extractInterfaces(implementation), creator = instantiator(implementation))
600
601 /**
602 * Java API: Uses the supplied Creator as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/ActorCell.scala, line 443 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: emptyBehaviorStack
Enclosing Method: ActorCell()
File: src/main/scala/akka/actor/ActorCell.scala:443
Taint Flags:

```
440 private[this] var _actor: Actor = _
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorCell.scala, line 443 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
441 def actor: Actor = _actor
442 var currentMessage: Envelope = _
443 private var behaviorStack: List[Actor.Receive] = emptyBehaviorStack
444 private[this] var sysmsgStash: LatestFirstSystemMessageList = SystemMessageList.LNil
445
446 // Java API
```

src/main/scala/akka/actor/ActorRef.scala, line 360 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: newActorCell
Enclosing Method: LocalActorRef()
File: src/main/scala/akka/actor/ActorRef.scala:360
Taint Flags:

```
357 * actorCell before we call init and start, since we can start using "this"
358 * object from another thread as soon as we run init.
359 */
360 private val actorCell: ActorCell = newActorCell(_system, this, _props, _dispatcher, _supervisor)
361 actorCell.init(sendSupervise = true, _mailboxType)
362
363 protected def newActorCell(
```

src/main/scala/akka/actor/Deployer.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$7
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:82
Taint Flags:

```
79 /**
80 * Java API to create a Deploy with the given RouterConfig
81 */
82 def this(routing: RouterConfig) = this("", ConfigFactory.empty, routing)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/Deployer.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
83
84 /**
85 * Java API to create a Deploy with the given RouterConfig with Scope
```

src/main/scala/akka/actor/TypedActor.scala, line 617 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:617
Taint Flags:

```
614 * appended in the sequence of interfaces.
615 */
616 def this(interface: Class[_ >: T], implementation: Class[T]) =
617 this(interfaces = TypedProps.extractInterfaces(interface), creator = instantiator(implementation))
618
619 /**
620 * Returns a new TypedProps with the specified dispatcher set.
```

src/main/scala/akka/actor/TypedActor.scala, line 617 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:617
Taint Flags:

```
614 * appended in the sequence of interfaces.
615 */
616 def this(interface: Class[_ >: T], implementation: Class[T]) =
617 this(interfaces = TypedProps.extractInterfaces(interface), creator = instantiator(implementation))
618
619 /**
620 * Returns a new TypedProps with the specified dispatcher set.
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 231 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: stoppingDecider
Enclosing Method: SupervisorStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:231
Taint Flags:

```
228 def stoppingDecider: Decider = {
229 case _: Exception => Stop
230 }
231 OneForOneStrategy()(stoppingDecider)
232 }
233
234 /**
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 181 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: TickDuration
Enclosing Method: LightArrayRevolverScheduler()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:181
Taint Flags:

```
178 }
179 }
180
181 override val maxFrequency: Double = 1.second / TickDuration
182
183 /*
184 * BELOW IS THE ACTUAL TIMER IMPLEMENTATION
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 188 (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.actor

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 188 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: TickDuration
Enclosing Method: LightArrayRevolverScheduler()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:188
Taint Flags:

```
185 */
186
187 private val start = clock()
188 private val tickNanos = TickDuration.toNanos
189 private val wheelMask = WheelSize - 1
190 private val queue = new TaskQueue
191
```

src/main/scala/akka/actor/ActorSystem.scala, line 958 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mailboxes
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:958
Taint Flags:

```
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
958 DefaultDispatcherPrerequisites(
959 threadFactory,
960 eventStream,
961 scheduler,
```

src/main/scala/akka/actor/ActorSystem.scala, line 822 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: _dynamicAccess
Enclosing Method: ActorSystemImpl()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 822 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/ActorSystem.scala:822
Taint Flags:

```
819 private val _dynamicAccess: DynamicAccess = createDynamicAccess()
820
821 final val settings: Settings = {
822 val config = Settings.amendSlf4jConfig(
823 applicationConfig.withFallback(ConfigFactory.defaultReference(classLoader)),
824 _dynamicAccess)
825 new Settings(classLoader, config, name, setup)
```

src/main/scala/akka/actor/ActorSystem.scala, line 933 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: logFilter
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:933
Taint Flags:

```
930 }
931
932 private[this] val markerLogging =
933 new MarkerLoggingAdapter(eventStream, getClass.getName + "(" + name + ")", this.getClass, logFilter)
934 val log: LoggingAdapter = markerLogging
935
936 val scheduler: Scheduler = createScheduler()
```

src/main/scala/akka/actor/FaultHandling.scala, line 639 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$1
Enclosing Method: OneForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:639
Taint Flags:

```
636 * Java API: Restart an infinite number of times. Compatible with lambda expressions.
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 639 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
637 */
638 def this(decider: SupervisorStrategy.Decider) =
639 this()(decider)
640
641 def withMaxNrOfRetries(maxNrOfRetries: Int): OneForOneStrategy = copy(maxNrOfRetries = maxNrOfRetries)(decider)
642
```

src/main/scala/akka/actor/TypedActor.scala, line 599 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:599
Taint Flags:

```
596 * appended in the sequence of interfaces.
597 */
598 def this(implementation: Class[T]) =
599 this(interfaces = TypedProps.extractInterfaces(implementation), creator = instantiator(implementation))
600
601 /**
602 * Java API: Uses the supplied Creator as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/TypedActor.scala, line 608 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:608
Taint Flags:

```
605 * appended in the sequence of interfaces.
606 */
607 def this(interface: Class[_ >: T], implementation: Creator[T]) =
608 this(interfaces = TypedProps.extractInterfaces(interface), creator = implementation.create _)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/TypedActor.scala, line 608 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
609
610 /**
611 * Java API: Uses the supplied class as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/TypedActor.scala, line 599 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:599
Taint Flags:

```
596 * appended in the sequence of interfaces.
597 */
598 def this(implementation: Class[T]) =
599 this(interfaces = TypedProps.extractInterfaces(implementation), creator = instantiator(implementation))
600
601 /**
602 * Java API: Uses the supplied Creator as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/Deployer.scala, line 21 (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: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:21
Taint Flags:

```
18 object Deploy {
19 final val NoDispatcherGiven = ""
20 final val NoMailboxGiven = ""
21 val local = Deploy(scope = LocalScope)
22
23 /**
24 * INTERNAL API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 500 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: AllForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:500
Taint Flags:

```
497 * Java API
498 */
499 def this(maxNrOfRetries: Int, withinTimeRange: Duration, trapExit: JIterable[Class[_ <: Throwable]]) =
500 this(maxNrOfRetries, withinTimeRange)(SupervisorStrategy.makeDecider(trapExit))
501
502 /**
503 * Java API
```

src/main/scala/akka/actor/ActorSystem.scala, line 954 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: deadLetters
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:954
Taint Flags:

```
951
952 def deadLetters: ActorRef = provider.deadLetters
953
954 val mailboxes: Mailboxes = new Mailboxes(settings, eventStream, dynamicAccess, deadLetters)
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
```

src/main/scala/akka/actor/Deployer.scala, line 21 (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.actor

src/main/scala/akka/actor/Deployer.scala, line 21 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: apply$default$2
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:21
Taint Flags:

```
18 object Deploy {
19  final val NoDispatcherGiven = ""
20  final val NoMailboxGiven = ""
21  val local = Deploy(scope = LocalScope)
22
23  /**
24  * INTERNAL API
```

src/main/scala/akka/actor/FaultHandling.scala, line 633 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$1
Enclosing Method: OneForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:633
Taint Flags:

```
630  this(maxNrOfRetries = maxNrOfRetries, withinTimeRange = withinTimeRange.asScala)(decider)
631
632  def this(loggingEnabled: Boolean, decider: SupervisorStrategy.Decider) =
633  this(loggingEnabled = loggingEnabled)(decider)
634
635  /**
636  * Java API: Restart an infinite number of times. Compatible with lambda expressions.
```

src/main/scala/akka/actor/FaultHandling.scala, line 506 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: AllForOneStrategy()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 506 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/FaultHandling.scala:506
Taint Flags:

```
503 * Java API
504 */
505 def this(maxNrOfRetries: Int, withinTimeRange: java.time.Duration, trapExit: JIterable[Class[_ <: Throwable]]) =
506 this(maxNrOfRetries, withinTimeRange.asScala)(SupervisorStrategy.makeDecider(trapExit))
507
508 /**
509 * Java API: compatible with lambda expressions
```

src/main/scala/akka/actor/TypedActor.scala, line 617 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:617
Taint Flags:

```
614 * appended in the sequence of interfaces.
615 */
616 def this(interface: Class[_ >: T], implementation: Class[T]) =
617 this(interfaces = TypedProps.extractInterfaces(interface), creator = instantiator(implementation))
618
619 /**
620 * Returns a new TypedProps with the specified dispatcher set.
```

src/main/scala/akka/actor/FaultHandling.scala, line 530 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$1
Enclosing Method: AllForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:530
Taint Flags:

```
527 * Java API: compatible with lambda expressions
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 530 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
528 */
529 def this(decider: SupervisorStrategy.Decider) =
530 this()(decider)
531
532 /*
533 * this is a performance optimization to avoid re-allocating the pairs upon
```

src/main/scala/akka/actor/Deployer.scala, line 87 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$7
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:87
Taint Flags:

```
84 /**
85 * Java API to create a Deploy with the given RouterConfig with Scope
86 */
87 def this(routing: RouterConfig, scope: Scope) = this("", ConfigFactory.empty, routing, scope)
88
89 /**
90 * Java API to create a Deploy with the given Scope
```

src/main/scala/akka/actor/ActorSystem.scala, line 958 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: scheduler
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:958
Taint Flags:

```
955
956 val dispatchers: Dispatchers = new Dispatchers(
957 settings,
958 DefaultDispatcherPrerequisites(
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 958 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
959 threadFactory,
960 eventStream,
961 scheduler,
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 187 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: clock
Enclosing Method: LightArrayRevolverScheduler()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:187
Taint Flags:

```
184 * BELOW IS THE ACTUAL TIMER IMPLEMENTATION
185 */
186
187 private val start = clock()
188 private val tickNanos = TickDuration.toNanos
189 private val wheelMask = WheelSize - 1
190 private val queue = new TaskQueue
```

src/main/scala/akka/actor/Props.scala, line 42 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply
Enclosing Method: Props()
File: src/main/scala/akka/actor/Props.scala:42
Taint Flags:

```
39 /**
40 * A Props instance whose creator will create an actor that doesn't respond to any message
41 */
42 final val empty = Props[EmptyActor]()
43
44 /**
45 * The default Props instance, uses the settings from the Props object starting with default*.
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 220 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: defaultDecider
Enclosing Method: SupervisorStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:220
Taint Flags:

```
217 * [[#defaultDecider]].
218 */
219 final val defaultStrategy: SupervisorStrategy = {
220 OneForOneStrategy()(defaultDecider)
221 }
222
223 /**
```

src/main/scala/akka/actor/Deployer.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:92
Taint Flags:

```
89 /**
90 * Java API to create a Deploy with the given Scope
91 */
92 def this(scope: Scope) = this("", ConfigFactory.empty, NoRouter, scope)
93
94 /**
95 * Do a merge between this and the other Deploy, where values from “this” take
```

src/main/scala/akka/actor/TypedActor.scala, line 599 (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.actor

src/main/scala/akka/actor/TypedActor.scala, line 599 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: extractInterfaces
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:599
Taint Flags:

```
596 * appended in the sequence of interfaces.
597 */
598 def this(implementation: Class[T]) =
599 this(interfaces = TypedProps.extractInterfaces(implementation), creator = instantiator(implementation))
600
601 /**
602 * Java API: Uses the supplied Creator as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/Deployer.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: Deploy()
File: src/main/scala/akka/actor/Deployer.scala:82
Taint Flags:

```
79 /**
80 * Java API to create a Deploy with the given RouterConfig
81 */
82 def this(routing: RouterConfig) = this("", ConfigFactory.empty, routing)
83
84 /**
85 * Java API to create a Deploy with the given RouterConfig with Scope
```

src/main/scala/akka/actor/FaultHandling.scala, line 639 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: OneForOneStrategy()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 639 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/actor/FaultHandling.scala:639
Taint Flags:

```
636 * Java API: Restart an infinite number of times. Compatible with lambda expressions.
637 */
638 def this(decider: SupervisorStrategy.Decider) =
639 this()(decider)
640
641 def withMaxNrOfRetries(maxNrOfRetries: Int): OneForOneStrategy = copy(maxNrOfRetries = maxNrOfRetries)(decider)
642
```

src/main/scala/akka/actor/TypedActor.scala, line 617 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: extractInterfaces
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:617
Taint Flags:

```
614 * appended in the sequence of interfaces.
615 */
616 def this(interface: Class[_ >: T], implementation: Class[T]) =
617 this(interfaces = TypedProps.extractInterfaces(interface), creator = instantiator(implementation))
618
619 /**
620 * Returns a new TypedProps with the specified dispatcher set.
```

src/main/scala/akka/actor/FaultHandling.scala, line 630 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: OneForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:630
Taint Flags:

```
627 * Java API: compatible with lambda expressions
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/FaultHandling.scala, line 630 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
628 */
629 def this(maxNrOfRetries: Int, withinTimeRange: java.time.Duration, decider: SupervisorStrategy.Decider) =
630 this(maxNrOfRetries = maxNrOfRetries, withinTimeRange = withinTimeRange.asScala)(decider)
631
632 def this(loggingEnabled: Boolean, decider: SupervisorStrategy.Decider) =
633 this(loggingEnabled = loggingEnabled)(decider)
```

src/main/scala/akka/actor/FaultHandling.scala, line 639 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: OneForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:639
Taint Flags:

```
636 * Java API: Restart an infinite number of times. Compatible with lambda expressions.
637 */
638 def this(decider: SupervisorStrategy.Decider) =
639 this()(decider)
640
641 def withMaxNrOfRetries(maxNrOfRetries: Int): OneForOneStrategy = copy(maxNrOfRetries = maxNrOfRetries)(decider)
642
```

src/main/scala/akka/actor/ActorPath.scala, line 287 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: toString
Enclosing Method: RootActorPath()
File: src/main/scala/akka/actor/ActorPath.scala:287
Taint Flags:

```
284
285 override val toString: String = address.toString + name
286
287 override val toSerializationFormat: String = toString
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

src/main/scala/akka/actor/ActorPath.scala, line 287 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
288
289 override def toStringWithAddress(addr: Address): String =
290 if (address.host.isDefined) address.toString + name
```

src/main/scala/akka/actor/TypedActor.scala, line 608 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: extractInterfaces
Enclosing Method: TypedProps()
File: src/main/scala/akka/actor/TypedActor.scala:608
Taint Flags:

```
605 * appended in the sequence of interfaces.
606 */
607 def this(interface: Class[_ >: T], implementation: Creator[T]) =
608 this(interfaces = TypedProps.extractInterfaces(interface), creator = implementation.create _)
609
610 /**
611 * Java API: Uses the supplied class as the factory for the TypedActor implementation,
```

src/main/scala/akka/actor/FaultHandling.scala, line 600 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: OneForOneStrategy()
File: src/main/scala/akka/actor/FaultHandling.scala:600
Taint Flags:

```
597 * Java API
598 */
599 def this(maxNrOfRetries: Int, withinTimeRange: Duration, decider: SupervisorStrategy.JDecider) =
600 this(maxNrOfRetries, withinTimeRange)(SupervisorStrategy.makeDecider(decider))
601
602 /**
603 * Java API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/ReceiveTimeout.scala, line 23 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: emptyReceiveTimeoutData
Enclosing Method: ReceiveTimeout()
File: src/main/scala/akka/actor/dungeon/ReceiveTimeout.scala:23
Taint Flags:

```
20 import ActorCell._
21 import ReceiveTimeout._
22
23 private var receiveTimeoutData: (Duration, Cancellable) = emptyReceiveTimeoutData
24
25 final def receiveTimeout: Duration = receiveTimeoutData._1
26
```

Package: akka.dispatch

src/main/scala/akka/dispatch/Dispatchers.scala, line 322 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: amendConfig
Enclosing Method: BalancingDispatcherConfigurator()
File: src/main/scala/akka/dispatch/Dispatchers.scala:322
Taint Flags:

```
319 */
320 @nowarn("msg=deprecated")
321 class BalancingDispatcherConfigurator(_config: Config, _prerequisites: DispatcherPrerequisites)
322  extends MessageDispatcherConfigurator(BalancingDispatcherConfigurator.amendConfig(_config), _prerequisites) {
323
324  private val instance = {
325  val mailboxes = prerequisites.mailboxes
```

src/main/scala/akka/dispatch/Dispatchers.scala, line 112 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Dispatchers.scala, line 112 (Code Correctness: Constructor Invokes Overridable Function) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: idConfig
Enclosing Method: Dispatchers()
File: src/main/scala/akka/dispatch/Dispatchers.scala:112
Taint Flags:

```
109
110 val cachingConfig = new CachingConfig(settings.config)
111
112 val defaultDispatcherConfig: Config =
113 idConfig(DefaultDispatcherId).withFallback(settings.config.getConfig(DefaultDispatcherId))
114
115 /**
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 285 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: stashCapacityFromConfig
Enclosing Method: Mailboxes()
File: src/main/scala/akka/dispatch/Mailboxes.scala:285
Taint Flags:

```
282 }
283
284 private val stashCapacityCache = new AtomicReference[Map[String, Int]](Map.empty[String, Int])
285 private val defaultStashCapacity: Int =
286 stashCapacityFromConfig(Dispatchers.DefaultDispatcherId, Mailboxes.DefaultMailboxId)
287
288 /**
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 396 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 396 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink: FunctionCall: createThreadPoolConfigBuilder
Enclosing Method: ThreadPoolExecutorConfigurator()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:396
Taint Flags:

```
393 class ThreadPoolExecutorConfigurator(config: Config, prerequisites: DispatcherPrerequisites)
394 extends ExecutorServiceConfigurator(config, prerequisites) {
395
396 val threadPoolConfig: ThreadPoolConfig = createThreadPoolConfigBuilder(config, prerequisites).config
397
398 protected def createThreadPoolConfigBuilder(
399 config: Config,
```

src/main/scala/akka/dispatch/Dispatchers.scala, line 125 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: lookup
Enclosing Method: Dispatchers()
File: src/main/scala/akka/dispatch/Dispatchers.scala:125
Taint Flags:

```
122 /**
123 * INTERNAL API
124 */
125 private[akka] val internalDispatcher = lookup(Dispatchers.InternalDispatcherId)
126
127 /**
128 * Returns a dispatcher as specified in configuration. Please note that this
```

src/main/scala/akka/dispatch/CachingConfig.scala, line 48 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: CachingConfig()
File: src/main/scala/akka/dispatch/CachingConfig.scala:48
Taint Flags:

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/CachingConfig.scala, line 48 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
45 import CachingConfig._
46
47 private val (config: Config, entryMap: ConcurrentHashMap[String, PathEntry]) = _config match {
48 case cc: CachingConfig => (cc.config, cc.entryMap)
49 case _ => (_config, new ConcurrentHashMap[String, PathEntry])
50 }
51
```

src/main/scala/akka/dispatch/CachingConfig.scala, line 48 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: entryMap
Enclosing Method: CachingConfig()
File: src/main/scala/akka/dispatch/CachingConfig.scala:48
Taint Flags:

```
45 import CachingConfig._
46
47 private val (config: Config, entryMap: ConcurrentHashMap[String, PathEntry]) = _config match {
48 case cc: CachingConfig => (cc.config, cc.entryMap)
49 case _ => (_config, new ConcurrentHashMap[String, PathEntry])
50 }
51
```

src/main/scala/akka/dispatch/Dispatchers.scala, line 324 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: create
Enclosing Method: BalancingDispatcherConfigurator()
File: src/main/scala/akka/dispatch/Dispatchers.scala:324
Taint Flags:

```
321 class BalancingDispatcherConfigurator(_config: Config, _prerequisites: DispatcherPrerequisites)
322 extends MessageDispatcherConfigurator(BalancingDispatcherConfigurator.amendConfig(_config), _prerequisites) {
323
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Dispatchers.scala, line 324 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
324 private val instance = {
325 val mailboxes = prerequisites.mailboxes
326 val id = config.getString("id")
327 val requirement = mailboxes.getMailboxRequirement(config)
```

Package: akka.dispatch.affinity

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 325 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akka$dispatch$affinity$AffinityPoolConfigurator$$rejectionHandlerFactoryFCQN
Enclosing Method: AffinityPoolConfigurator()
File: src/main/scala/akka/dispatch/affinity/AffinityPool.scala:325
Taint Flags:

```
322 .get
323
324 private val rejectionHandlerFactoryFCQN = config.getString("rejection-handler")
325 private val rejectionHandlerFactory = prerequisites.dynamicAccess
326 .createInstanceFor[RejectionHandlerFactory](rejectionHandlerFactoryFCQN, Nil)
327 .recover {
328 case exception =>
```

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 136 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bookKeepingLock
Enclosing Method: AffinityPool()
File: src/main/scala/akka/dispatch/affinity/AffinityPool.scala:136
Taint Flags:

```
133 private val bookKeepingLock = new ReentrantGuard()
134
135 // condition used for awaiting termination
136 private val terminationCondition = bookKeepingLock.newCondition()
137
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.dispatch.affinity

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 136 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
138 // indicates the current state of the pool
139 @volatile final private var poolState: PoolState = Uninitialized
```

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 314 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akka$dispatch$affinity$AffinityPoolConfigurator$$queueSelectorFactoryFQCN
Enclosing Method: AffinityPoolConfigurator()
File: src/main/scala/akka/dispatch/affinity/AffinityPool.scala:314
Taint Flags:

```
311 .requiring(level => 1 <= level && level <= 10, "idle-cpu-level must be between 1 and 10")
312
313 private val queueSelectorFactoryFQCN = config.getString("queue-selector")
314 private val queueSelectorFactory: QueueSelectorFactory =
315 prerequisites.dynamicAccess
316 .createInstanceFor[QueueSelectorFactory](queueSelectorFactoryFQCN, immutable.Seq(classOf[Config] -> config))
317 .recover {
```

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 540 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: DebugLevel
Enclosing Method: Logging()
File: src/main/scala/akka/event/Logging.scala:540
Taint Flags:

```
537 }
538
539 // these type ascriptions/casts are necessary to avoid CCEs during construction while retaining correct type
540 val AllLogLevels: immutable.Seq[LogLevel] = Vector(ErrorLevel, WarningLevel, InfoLevel, DebugLevel)
541
542 /**
543 * Obtain LoggingAdapter for the given actor system and source object. This
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 540 (Code Correctness: Constructor Invokes Overridable Function)	Low

src/main/scala/akka/event/Logging.scala, line 318 (Code Correctness: Constructor Invokes Overridable Function)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: LogSource$$anon$4
Enclosing Method: LogSource()
File: src/main/scala/akka/event/Logging.scala:318
Taint Flags:

```
315 override def genString(a: Actor, system: ActorSystem) = fromActorRef.genString(a.self, system)
316 }
317
318 implicit val fromActorRef: LogSource[ActorRef] = new LogSource[ActorRef] {
319 def genString(a: ActorRef) = a.path.toString
320 override def genString(a: ActorRef, system: ActorSystem) =
321 try {
```

src/main/scala/akka/event/Logging.scala, line 340 (Code Correctness: Constructor Invokes Overridable Function)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: LogSource$$anon$6
Enclosing Method: LogSource()
File: src/main/scala/akka/event/Logging.scala:340
Taint Flags:

```
337 }
338
339 // this one unfortunately does not work as implicit, because existential types have some weird behavior
340 val fromClass: LogSource[Class[_]] = new LogSource[Class[_]] {
341 def genString(c: Class[_]): String = Logging.simpleName(c)
342 override def genString(c: Class[_], system: ActorSystem): String = genString(c) + "(" + system + ")"
343 override def getClazz(c: Class[_]): Class[_] = c
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 540 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: InfoLevel
Enclosing Method: Logging()
File: src/main/scala/akka/event/Logging.scala:540
Taint Flags:

```
537 }
538
539 // these type ascriptions/casts are necessary to avoid CCEs during construction while retaining correct type
540 val AllLogLevels: immutable.Seq[LogLevel] = Vector(ErrorLevel, WarningLevel, InfoLevel, DebugLevel)
541
542 /**
543 * Obtain LoggingAdapter for the given actor system and source object. This
```

src/main/scala/akka/event/DeadLetterListener.scala, line 27 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: maxCount
Enclosing Method: DeadLetterListener()
File: src/main/scala/akka/event/DeadLetterListener.scala:27
Taint Flags:

```
24
25 val eventStream: EventStream = context.system.eventStream
26 protected val maxCount: Int = context.system.settings.LogDeadLetters
27 private val isAlwaysLoggingDeadLetters = maxCount == Int.MaxValue
28 protected var count: Int = 0
29
30 override def preStart(): Unit = {
```

src/main/scala/akka/event/Logging.scala, line 1672 (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.event

src/main/scala/akka/event/Logging.scala, line 1672 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: apply
Enclosing Method: LogMarker()
File: src/main/scala/akka/event/Logging.scala:1672
Taint Flags:

```
1669 case None => None
1670 }
1671
1672 private[akka] final val Security = apply("SECURITY")
1673
1674 /**
1675 * INTERNAL API
```

src/main/scala/akka/event/Logging.scala, line 307 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: LogSource$$anon$2
Enclosing Method: LogSource()
File: src/main/scala/akka/event/Logging.scala:307
Taint Flags:

```
304 * </ul>
305 */
306 object LogSource {
307 implicit val fromString: LogSource[String] = new LogSource[String] {
308 def genString(s: String) = s
309 override def genString(s: String, system: ActorSystem) = s + "(" + system + ")"
310 override def getClazz(s: String) = classOf[DummyClassForStringSources]
```

src/main/scala/akka/event/Logging.scala, line 333 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: LogSource$$anon$5
Enclosing Method: LogSource()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 333 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/event/Logging.scala:333
Taint Flags:

```
330 * INTERNAL API
331 */
332 @InternalApi private[akka] implicit val fromActorWithLoggerClass: LogSource[ActorWithLogClass] =
333 new LogSource[ActorWithLogClass] {
334 def genString(a: ActorWithLogClass) = fromActor.genString(a.actor)
335 override def genString(a: ActorWithLogClass, system: ActorSystem) = fromActor.genString(a.actor, system)
336 override def getClazz(a: ActorWithLogClass): Class[_] = a.logClass
```

src/main/scala/akka/event/Logging.scala, line 540 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: ErrorLevel
Enclosing Method: Logging()
File: src/main/scala/akka/event/Logging.scala:540
Taint Flags:

```
537 }
538
539 // these type ascriptions/casts are necessary to avoid CCEs during construction while retaining correct type
540 val AllLogLevels: immutable.Seq[LogLevel] = Vector(ErrorLevel, WarningLevel, InfoLevel, DebugLevel)
541
542 /**
543 * Obtain LoggingAdapter for the given actor system and source object. This
```

src/main/scala/akka/event/Logging.scala, line 313 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: LogSource$$anon$3
Enclosing Method: LogSource()
File: src/main/scala/akka/event/Logging.scala:313
Taint Flags:

```
310 override def getClazz(s: String) = classOf[DummyClassForStringSources]
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 313 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
311 }
312
313 implicit val fromActor: LogSource[Actor] = new LogSource[Actor] {
314 def genString(a: Actor) = fromActorRef.genString(a.self)
315 override def genString(a: Actor, system: ActorSystem) = fromActorRef.genString(a.self, system)
316 }
```

src/main/scala/akka/event/Logging.scala, line 540 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: WarningLevel
Enclosing Method: Logging()
File: src/main/scala/akka/event/Logging.scala:540
Taint Flags:

```
537 }
538
539 // these type ascriptions/casts are necessary to avoid CCEs during construction while retaining correct type
540 val AllLogLevels: immutable.Seq[LogLevel] = Vector(ErrorLevel, WarningLevel, InfoLevel, DebugLevel)
541
542 /**
543 * Obtain LoggingAdapter for the given actor system and source object. This
```

Package: akka.io

src/main/scala/akka/io/SimpleDnsManager.scala, line 34 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akkaioSimpleDnsManager$$cacheCleanup
Enclosing Method: SimpleDnsManager()
File: src/main/scala/akka/io/SimpleDnsManager.scala:34
Taint Flags:

```
31 case _ => None
32 }
33
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io

src/main/scala/akka/io/SimpleDnsManager.scala, line 34 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
34 private val cleanupTimer = cacheCleanup.map { _ =>
35 val interval = Duration(
36 ext.Settings.ResolverConfig.getDuration("cache-cleanup-interval", TimeUnit.MILLISECONDS),
37 TimeUnit.MILLISECONDS)
```

src/main/scala/akka/io/UdpConnection.scala, line 48 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akkaioUdpConnection$$reportConnectFailure
Enclosing Method: UdpConnection()
File: src/main/scala/akka/io/UdpConnection.scala:48
Taint Flags:

```
45 if (remoteAddress.isUnresolved) {
46 Dns.resolve(DnsProtocol.Resolve(remoteAddress.getHostName), context.system, self) match {
47 case Some(r) =>
48 reportConnectFailure {
49 doConnect(new InetSocketAddress(r.address(), remoteAddress.getPort))
50 }
51 case None =>
```

src/main/scala/akka/io/UdpConnection.scala, line 55 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akkaioUdpConnection$$reportConnectFailure
Enclosing Method: UdpConnection()
File: src/main/scala/akka/io/UdpConnection.scala:55
Taint Flags:

```
52 context.become(resolving())
53 }
54 } else {
55 reportConnectFailure {
56 doConnect(remoteAddress)
57 }
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.io

src/main/scala/akka/io/UdpConnection.scala, line 55 (Code Correctness: Constructor Invokes Overridable Function) **Low**

```
58 }
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 100 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: getTtl
Enclosing Method: InetAddressDnsResolver()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:100
Taint Flags:

```
97  }
98  }
99
100 val positiveCachePolicy: CachePolicy = getTtl("positive-ttl", positive = true)
101 val negativeCachePolicy: CachePolicy = getTtl("negative-ttl", positive = false)
102 @deprecated("Use positiveCacheDuration instead", "2.5.17")
103 val positiveTtl: Long = toLongTtl(positiveCachePolicy)
```

src/main/scala/akka/io/TcpListener.scala, line 52 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: channel
Enclosing Method: TcpListener()
File: src/main/scala/akka/io/TcpListener.scala:52
Taint Flags:

```
49 context.watch(bind.handler) // sign death pact
50
51 val channel = ServerSocketChannel.open
52 channel.configureBlocking(false)
53
54 var acceptLimit = if (bind.pullMode) 0 else BatchAcceptLimit
55
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io

src/main/scala/akka/io/Tcp.scala, line 645 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: Settings
Enclosing Method: TcpExt()
File: src/main/scala/akka/io/Tcp.scala:645
Taint Flags:

```
642 /**
643 *
644 */
645 val manager: ActorRef = {
646 system.systemActorOf(
647 props = Props(classOf[TcpManager], this).withDispatcher(Settings.ManagementDispatcher).withDeploy(Deploy.local),
648 name = "IO-TCP")
```

src/main/scala/akka/io/Tcp.scala, line 656 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: Settings
Enclosing Method: TcpExt()
File: src/main/scala/akka/io/Tcp.scala:656
Taint Flags:

```
653 */
654 def getManager: ActorRef = manager
655
656 val bufferPool: BufferPool = new DirectByteBufferPool(Settings.DirectBufferSize, Settings.MaxDirectBufferPoolSize)
657 val fileIoDispatcher = system.dispatchers.lookup(Settings.FileIODispatcher)
658 }
659
```

src/main/scala/akka/io/Tcp.scala, line 656 (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.io

src/main/scala/akka/io/Tcp.scala, line 656 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: Settings
Enclosing Method: TcpExt()
File: src/main/scala/akka/io/Tcp.scala:656
Taint Flags:

```
653 */
654 def getManager: ActorRef = manager
655
656 val bufferPool: BufferPool = new DirectByteBufferPool(Settings.DirectBufferSize, Settings.MaxDirectBufferPoolSize)
657 val fileIoDispatcher = system.dispatchers.lookup(Settings.FileIODispatcher)
658 }
659
```

src/main/scala/akka/io/Tcp.scala, line 657 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: Settings
Enclosing Method: TcpExt()
File: src/main/scala/akka/io/Tcp.scala:657
Taint Flags:

```
654 def getManager: ActorRef = manager
655
656 val bufferPool: BufferPool = new DirectByteBufferPool(Settings.DirectBufferSize, Settings.MaxDirectBufferPoolSize)
657 val fileIoDispatcher = system.dispatchers.lookup(Settings.FileIODispatcher)
658 }
659
660 /**
```

src/main/scala/akka/io/UdpSender.scala, line 43 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: channel
Enclosing Method: UdpSender()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io

src/main/scala/akka/io/UdpSender.scala, line 43 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/io/UdpSender.scala:43
Taint Flags:

```
40
41 datagramChannel
42 }
43 channelRegistry.register(channel, initialOps = 0)
44
45 def receive: Receive = {
46 case registration: ChannelRegistration =>
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 105 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: toLongTtl
Enclosing Method: InetAddressDnsResolver()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:105
Taint Flags:

```
102 @deprecated("Use positiveCacheDuration instead", "2.5.17")
103 val positiveTtl: Long = toLongTtl(positiveCachePolicy)
104 @deprecated("Use negativeCacheDuration instead", "2.5.17")
105 val negativeTtl: Long = toLongTtl(negativeCachePolicy)
106
107 private def toLongTtl(cp: CachePolicy): Long = {
108 cp match {
```

src/main/scala/akka/io/Udp.scala, line 226 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: UdpExt()
File: src/main/scala/akka/io/Udp.scala:226
Taint Flags:

```
223
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io

src/main/scala/akka/io/Udp.scala, line 226 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
224 val settings: UdpSettings = new UdpSettings(system.settings.config.getConfig("akka.io.udp"))
225
226 val manager: ActorRef = {
227 system.systemActorOf(
228 props = Props(classOf[UdpManager], this).withDispatcher(settings.ManagementDispatcher).withDeploy(Deploy.local),
229 name = "IO-UDP-FF")
```

src/main/scala/akka/io/Udp.scala, line 241 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: UdpExt()
File: src/main/scala/akka/io/Udp.scala:241
Taint Flags:

```
238 * INTERNAL API
239 */
240 private[io] val bufferPool: BufferPool =
241 new DirectByteBufferPool(settings.DirectBufferSize, settings.MaxDirectBufferPoolSize)
242 }
243
244 /**
```

src/main/scala/akka/io/Udp.scala, line 241 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: UdpExt()
File: src/main/scala/akka/io/Udp.scala:241
Taint Flags:

```
238 * INTERNAL API
239 */
240 private[io] val bufferPool: BufferPool =
241 new DirectByteBufferPool(settings.DirectBufferSize, settings.MaxDirectBufferPoolSize)
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.io

src/main/scala/akka/io/Udp.scala, line 241 (Code Correctness: Constructor Invokes Overridable Function) **Low**

```
242 }
243
244 /**
```

src/main/scala/akka/io/Dns.scala, line 224 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: Settings
Enclosing Method: DnsExt()
File: src/main/scala/akka/io/Dns.scala:224
Taint Flags:

```
221
222 // System DNS resolver
223 @nowarn("msg=deprecated")
224 val provider: DnsProvider =
225 system.dynamicAccess.createInstanceFor[DnsProvider](Settings.ProviderObjectName, Nil).get
226
227 // System DNS cache
```

src/main/scala/akka/io/Dns.scala, line 231 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: Settings
Enclosing Method: DnsExt()
File: src/main/scala/akka/io/Dns.scala:231
Taint Flags:

```
228 val cache: Dns = provider.cache
229
230 // System DNS manager
231 val manager: ActorRef = {
232 system.systemActorOf(
233 props = Props(provider.managerClass, this).withDeploy(Deploy.local).withDispatcher(Settings.Dispatcher),
234 name = managerName)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 101 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: getTtl
Enclosing Method: InetAddressDnsResolver()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:101
Taint Flags:

```
98  }
99
100 val positiveCachePolicy: CachePolicy = getTtl("positive-ttl", positive = true)
101 val negativeCachePolicy: CachePolicy = getTtl("negative-ttl", positive = false)
102 @deprecated("Use positiveCacheDuration instead", "2.5.17")
103 val positiveTtl: Long = toLongTtl(positiveCachePolicy)
104 @deprecated("Use negativeCacheDuration instead", "2.5.17")
```

src/main/scala/akka/io/SimpleDnsCache.scala, line 31 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: expiryEntryOrdering
Enclosing Method: SimpleDnsCache()
File: src/main/scala/akka/io/SimpleDnsCache.scala:31
Taint Flags:

```
28 class SimpleDnsCache extends Dns with PeriodicCacheCleanup with NoSerializationVerificationNeeded {
29  import SimpleDnsCache._
30  private val cacheRef = new AtomicReference(
31  new Cache[(String, RequestType), Resolved](
32  immutable.SortedSet()(expiryEntryOrdering()),
33  immutable.Map(),
34  () => clock()))
```

src/main/scala/akka/io/UdpConnected.scala, line 164 (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.io

src/main/scala/akka/io/UdpConnected.scala, line 164 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: settings
Enclosing Method: UdpConnectedExt()
File: src/main/scala/akka/io/UdpConnected.scala:164
Taint Flags:

```
161
162 val settings: UdpSettings = new UdpSettings(system.settings.config.getConfig("akka.io.udp-connected"))
163
164 val manager: ActorRef = {
165 system.systemActorOf(
166 props = Props(classOf[UdpConnectedManager], this)
167 .withDispatcher(settings.ManagementDispatcher)
```

src/main/scala/akka/io/UdpConnected.scala, line 177 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: UdpConnectedExt()
File: src/main/scala/akka/io/UdpConnected.scala:177
Taint Flags:

```
174 */
175 def getManager: ActorRef = manager
176
177 val bufferPool: BufferPool = new DirectByteBufferPool(settings.DirectBufferSize, settings.MaxDirectBufferPoolSize)
178
179 }
180
```

src/main/scala/akka/io/UdpConnected.scala, line 177 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: UdpConnectedExt()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io

src/main/scala/akka/io/UdpConnected.scala, line 177 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/io/UdpConnected.scala:177
Taint Flags:

```
174 */
175 def getManager: ActorRef = manager
176
177 val bufferPool: BufferPool = new DirectByteBufferPool(settings.DirectBufferSize, settings.MaxDirectBufferPoolSize)
178
179 }
180
```

src/main/scala/akka/io/UdpConnection.scala, line 52 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: resolving
Enclosing Method: UdpConnection()
File: src/main/scala/akka/io/UdpConnection.scala:52
Taint Flags:

```
49 doConnect(new InetSocketAddress(r.address(), remoteAddress.getPort))
50 }
51 case None =>
52 context.become(resolving())
53 }
54 } else {
55 reportConnectFailure {
```

src/main/scala/akka/io/TcpListener.scala, line 56 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: liftedTree1
Enclosing Method: TcpListener()
File: src/main/scala/akka/io/TcpListener.scala:56
Taint Flags:

```
53
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io

src/main/scala/akka/io/TcpListener.scala, line 56 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
54 var acceptLimit = if (bind.pullMode) 0 else BatchAcceptLimit
55
56 val localAddress =
57 try {
58 val socket = channel.socket
59 bind.options.foreach(_.beforeServerSocketBind(socket))
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 103 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: toLongTtl
Enclosing Method: InetAddressDnsResolver()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:103
Taint Flags:

```
100 val positiveCachePolicy: CachePolicy = getTtl("positive-ttl", positive = true)
101 val negativeCachePolicy: CachePolicy = getTtl("negative-ttl", positive = false)
102 @deprecated("Use positiveCacheDuration instead", "2.5.17")
103 val positiveTtl: Long = toLongTtl(positiveCachePolicy)
104 @deprecated("Use negativeCacheDuration instead", "2.5.17")
105 val negativeTtl: Long = toLongTtl(negativeCachePolicy)
106
```

src/main/scala/akka/io/Dns.scala, line 228 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: provider
Enclosing Method: DnsExt()
File: src/main/scala/akka/io/Dns.scala:228
Taint Flags:

```
225 system.dynamicAccess.createInstanceFor[DnsProvider](Settings.ProviderObjectName, Nil).get
226
227 // System DNS cache
228 val cache: Dns = provider.cache
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io

src/main/scala/akka/io/Dns.scala, line 228 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
229
230  // System DNS manager
231  val manager: ActorRef = {
```

src/main/scala/akka/io/Dns.scala, line 231 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: provider
Enclosing Method: DnsExt()
File: src/main/scala/akka/io/Dns.scala:231
Taint Flags:

```
228  val cache: Dns = provider.cache
229
230  // System DNS manager
231  val manager: ActorRef = {
232  system.systemActorOf(
233  props = Props(provider.managerClass, this).withDeploy(Deploy.local).withDispatcher(Settings.Dispatcher),
234  name = managerName)
```

src/main/scala/akka/io/DirectByteBufferPool.scala, line 81 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: liftedTree1
Enclosing Method: DirectByteBufferPool()
File: src/main/scala/akka/io/DirectByteBufferPool.scala:81
Taint Flags:

```
78
79 /** INTERNAL API */
80 private[akka] object DirectByteBufferPool {
81  private val CleanDirectBuffer: ByteBuffer => Unit =
82  try {
83  val cleanerMethod = Class.forName("java.nio.DirectByteBuffer").getMethod("cleaner")
84  cleanerMethod.setAccessible(true)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 103 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: positiveCachePolicy
Enclosing Method: InetAddressDnsResolver()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:103
Taint Flags:

```
100 val positiveCachePolicy: CachePolicy = getTtl("positive-ttl", positive = true)
101 val negativeCachePolicy: CachePolicy = getTtl("negative-ttl", positive = false)
102 @deprecated("Use positiveCacheDuration instead", "2.5.17")
103 val positiveTtl: Long = toLongTtl(positiveCachePolicy)
104 @deprecated("Use negativeCacheDuration instead", "2.5.17")
105 val negativeTtl: Long = toLongTtl(negativeCachePolicy)
106
```

src/main/scala/akka/io/UdpListener.scala, line 45 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: liftedTree1
Enclosing Method: UdpListener()
File: src/main/scala/akka/io/UdpListener.scala:45
Taint Flags:

```
42 .create()
43 channel.configureBlocking(false)
44
45 val localAddress =
46 try {
47 val socket = channel.socket
48 bind.options.foreach(_.beforeDatagramBind(socket))
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 105 (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.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 105 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: negativeCachePolicy
Enclosing Method: InetAddressDnsResolver()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:105
Taint Flags:

```
102 @deprecated("Use positiveCacheDuration instead", "2.5.17")
103 val positiveTtl: Long = toLongTtl(positiveCachePolicy)
104 @deprecated("Use negativeCacheDuration instead", "2.5.17")
105 val negativeTtl: Long = toLongTtl(negativeCachePolicy)
106
107 private def toLongTtl(cp: CachePolicy): Long = {
108 cp match {
```

src/main/scala/akka/io/UdpListener.scala, line 43 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: channel
Enclosing Method: UdpListener()
File: src/main/scala/akka/io/UdpListener.scala:43
Taint Flags:

```
40 }
41 .getOrElse(DatagramChannelCreator())
42 .create()
43 channel.configureBlocking(false)
44
45 val localAddress =
46 try {
```

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 58 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: getTtl

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 58 (Code Correctness: Constructor Invokes Overridable Function) — Low

Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:58
Taint Flags:

```
55 val ResolveTimeout: FiniteDuration = c.getDuration("resolve-timeout").asScala
56
57 val PositiveCachePolicy: CachePolicy = getTtl("positive-ttl")
58 val NegativeCachePolicy: CachePolicy = getTtl("negative-ttl")
59
60 private def getTtl(path: String): CachePolicy =
61 c.getString(path) match {
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 94 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: resolvConf
Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:94
Taint Flags:

```
91 c.getValue("search-domains").valueType() match {
92 case ConfigValueType.STRING =>
93 c.getString("search-domains") match {
94 case "default" => resolvConf.map(_.search).getOrElse(Nil)
95 case single => List(single)
96 }
97 case ConfigValueType.LIST =>
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 107 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: resolvConf
Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:107
Taint Flags:

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 107 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
104 c.getValue("ndots").valueType() match {
105 case ConfigValueType.STRING =>
106 c.getString("ndots") match {
107 case "default" => resolvConf.map(_.ndots).getOrElse(1)
108 case _ =>
109 throw new IllegalArgumentException("Invalid value for ndots. Must be the string 'default' or an integer.")
110 }
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: failUnableToDetermineDefaultNameservers
Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:41
Taint Flags:

```
38 c.getString("nameservers") match {
39 case "default" =>
40 val osAddresses = getDefaultNameServers(system).getOrElse(failUnableToDetermineDefaultNameservers)
41 if (osAddresses.isEmpty) failUnableToDetermineDefaultNameservers
42 osAddresses
43 case other =>
44 parseNameserverAddress(other) :: Nil
```

src/main/scala/akka/io/dns/RecordType.scala, line 34 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:34
Taint Flags:

```
31 }
32
33 /** A host address */
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 34 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
34 final val A: RecordType = register(RecordType(1, "A"))
35
36 /** An authoritative name server */
37 final val NS: RecordType = register(RecordType(2, "NS"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 37 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:37
Taint Flags:

```
34 final val A: RecordType = register(RecordType(1, "A"))
35
36 /** An authoritative name server */
37 final val NS: RecordType = register(RecordType(2, "NS"))
38
39 /** A mail destination (Obsolete - use MX) */
40 final val MD: RecordType = register(RecordType(3, "MD"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 40 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:40
Taint Flags:

```
37 final val NS: RecordType = register(RecordType(2, "NS"))
38
39 /** A mail destination (Obsolete - use MX) */
40 final val MD: RecordType = register(RecordType(3, "MD"))
41
42 /** A mail forwarder (Obsolete - use MX) */
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 40 (Code Correctness: Constructor Invokes Overridable Function) **Low**

```
43 final val MF: RecordType = register(RecordType(4, "MF"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 43 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:43
Taint Flags:

```
40 final val MD: RecordType = register(RecordType(3, "MD"))
41
42 /** A mail forwarder (Obsolete - use MX) */
43 final val MF: RecordType = register(RecordType(4, "MF"))
44
45 /** the canonical name for an alias */
46 final val CNAME: RecordType = register(RecordType(5, "CNAME"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 46 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:46
Taint Flags:

```
43 final val MF: RecordType = register(RecordType(4, "MF"))
44
45 /** the canonical name for an alias */
46 final val CNAME: RecordType = register(RecordType(5, "CNAME"))
47
48 /** marks the start of a zone of authority */
49 final val SOA: RecordType = register(RecordType(6, "SOA"))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 49 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:49
Taint Flags:

```
46 final val CNAME: RecordType = register(RecordType(5, "CNAME"))
47
48 /** marks the start of a zone of authority */
49 final val SOA: RecordType = register(RecordType(6, "SOA"))
50
51 /** A mailbox domain name (EXPERIMENTAL) */
52 final val MB: RecordType = register(RecordType(7, "MB"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 52 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:52
Taint Flags:

```
49 final val SOA: RecordType = register(RecordType(6, "SOA"))
50
51 /** A mailbox domain name (EXPERIMENTAL) */
52 final val MB: RecordType = register(RecordType(7, "MB"))
53
54 /** A mail group member (EXPERIMENTAL) */
55 final val MG: RecordType = register(RecordType(8, "MG"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 55 (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.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 55 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:55
Taint Flags:

```
52 final val MB: RecordType = register(RecordType(7, "MB"))
53
54 /** A mail group member (EXPERIMENTAL) */
55 final val MG: RecordType = register(RecordType(8, "MG"))
56
57 /** A mail rename domain name (EXPERIMENTAL) */
58 final val MR: RecordType = register(RecordType(9, "MR"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 58 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:58
Taint Flags:

```
55 final val MG: RecordType = register(RecordType(8, "MG"))
56
57 /** A mail rename domain name (EXPERIMENTAL) */
58 final val MR: RecordType = register(RecordType(9, "MR"))
59
60 /** A null RR (EXPERIMENTAL) */
61 final val NULL: RecordType = register(RecordType(10, "NULL"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 61 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 61 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/io/dns/RecordType.scala:61
Taint Flags:

```
58 final val MR: RecordType = register(RecordType(9, "MR"))
59
60 /** A null RR (EXPERIMENTAL) */
61 final val NULL: RecordType = register(RecordType(10, "NULL"))
62
63 /** A well known service description */
64 final val WKS: RecordType = register(RecordType(11, "WKS"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 64 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:64
Taint Flags:

```
61 final val NULL: RecordType = register(RecordType(10, "NULL"))
62
63 /** A well known service description */
64 final val WKS: RecordType = register(RecordType(11, "WKS"))
65
66 /** A domain name pointer */
67 final val PTR: RecordType = register(RecordType(12, "PTR"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 67 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:67
Taint Flags:

```
64 final val WKS: RecordType = register(RecordType(11, "WKS"))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 67 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
65
66 /** A domain name pointer */
67 final val PTR: RecordType = register(RecordType(12, "PTR"))
68
69 /** host information */
70 final val HINFO: RecordType = register(RecordType(13, "HINFO"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 70 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:70
Taint Flags:

```
67 final val PTR: RecordType = register(RecordType(12, "PTR"))
68
69 /** host information */
70 final val HINFO: RecordType = register(RecordType(13, "HINFO"))
71
72 /** mailbox or mail list information */
73 final val MINFO: RecordType = register(RecordType(14, "MINFO"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 73 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:73
Taint Flags:

```
70 final val HINFO: RecordType = register(RecordType(13, "HINFO"))
71
72 /** mailbox or mail list information */
73 final val MINFO: RecordType = register(RecordType(14, "MINFO"))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 73 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
74
75  /** mail exchange */
76  final val MX: RecordType = register(RecordType(15, "MX"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 76 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:76
Taint Flags:

```
73  final val MINFO: RecordType = register(RecordType(14, "MINFO"))
74
75  /** mail exchange */
76  final val MX: RecordType = register(RecordType(15, "MX"))
77
78  /** text strings */
79  final val TXT: RecordType = register(RecordType(16, "TXT"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 79 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:79
Taint Flags:

```
76  final val MX: RecordType = register(RecordType(15, "MX"))
77
78  /** text strings */
79  final val TXT: RecordType = register(RecordType(16, "TXT"))
80
81  /** The AAAA resource record type is a record specific to the Internet class that stores a single IPv6 address. */
82  // See: https://tools.ietf.org/html/rfc3596
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 83 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:83
Taint Flags:

```
80
81 /** The AAAA resource record type is a record specific to the Internet class that stores a single IPv6 address. */
82 // See: https://tools.ietf.org/html/rfc3596
83 final val AAAA: RecordType = register(RecordType(28, "AAAA"))
84
85 /**
86 * The SRV RR allows administrators to use several servers for a single
```

src/main/scala/akka/io/dns/RecordType.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:92
Taint Flags:

```
89 * backups.
90 */
91 // See: https://tools.ietf.org/html/rfc2782
92 final val SRV: RecordType = register(RecordType(33, "SRV"))
93
94 final val AXFR: RecordType = register(RecordType(252, "AXFR"))
95 final val MAILB: RecordType = register(RecordType(253, "MAILB"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 94 (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.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 94 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:94
Taint Flags:

```
91 // See: https://tools.ietf.org/html/rfc2782
92 final val SRV: RecordType = register(RecordType(33, "SRV"))
93
94 final val AXFR: RecordType = register(RecordType(252, "AXFR"))
95 final val MAILB: RecordType = register(RecordType(253, "MAILB"))
96 final val MAILA: RecordType = register(RecordType(254, "MAILA"))
97 final val WILDCARD: RecordType = register(RecordType(255, "WILDCARD"))
```

src/main/scala/akka/io/dns/RecordType.scala, line 95 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:95
Taint Flags:

```
92 final val SRV: RecordType = register(RecordType(33, "SRV"))
93
94 final val AXFR: RecordType = register(RecordType(252, "AXFR"))
95 final val MAILB: RecordType = register(RecordType(253, "MAILB"))
96 final val MAILA: RecordType = register(RecordType(254, "MAILA"))
97 final val WILDCARD: RecordType = register(RecordType(255, "WILDCARD"))
98 }
```

src/main/scala/akka/io/dns/RecordType.scala, line 96 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/RecordType.scala, line 96 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/io/dns/RecordType.scala:96
Taint Flags:

```
93
94  final val AXFR: RecordType = register(RecordType(252, "AXFR"))
95  final val MAILB: RecordType = register(RecordType(253, "MAILB"))
96  final val MAILA: RecordType = register(RecordType(254, "MAILA"))
97  final val WILDCARD: RecordType = register(RecordType(255, "WILDCARD"))
98 }
99
```

src/main/scala/akka/io/dns/RecordType.scala, line 97 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: register
Enclosing Method: RecordType()
File: src/main/scala/akka/io/dns/RecordType.scala:97
Taint Flags:

```
94  final val AXFR: RecordType = register(RecordType(252, "AXFR"))
95  final val MAILB: RecordType = register(RecordType(253, "MAILB"))
96  final val MAILA: RecordType = register(RecordType(254, "MAILA"))
97  final val WILDCARD: RecordType = register(RecordType(255, "WILDCARD"))
98 }
99
100 undefined
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 44 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: parseNameserverAddress
Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:44
Taint Flags:

```
41  if (osAddresses.isEmpty) failUnableToDetermineDefaultNameservers
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 44 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
42 osAddresses
43 case other =>
44 parseNameserverAddress(other) :: Nil
45 }
46 case ConfigValueType.LIST =>
47 val userAddresses =
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 40 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: getDefaultNameServers
Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:40
Taint Flags:

```
37 case ConfigValueType.STRING =>
38 c.getString("nameservers") match {
39 case "default" =>
40 val osAddresses = getDefaultNameServers(system).getOrElse(failUnableToDetermineDefaultNameservers)
41 if (osAddresses.isEmpty) failUnableToDetermineDefaultNameservers
42 osAddresses
43 case other =>
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 57 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: getTtl
Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:57
Taint Flags:

```
54
55 val ResolveTimeout: FiniteDuration = c.getDuration("resolve-timeout").asScala
56
57 val PositiveCachePolicy: CachePolicy = getTtl("positive-ttl")
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 57 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
58 val NegativeCachePolicy: CachePolicy = getTtl("negative-ttl")
59
60 private def getTtl(path: String): CachePolicy =
```

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 72 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nameServers
Enclosing Method: AsyncDnsResolver()
File: src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala:72
Taint Flags:

```
69
70 val positiveCachePolicy = settings.PositiveCachePolicy
71 val negativeCachePolicy = settings.NegativeCachePolicy
72 log.debug(
73 "Using name servers [{}] and search domains [{}] with ndots={}",
74 nameServers,
75 settings.SearchDomains,
```

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 85 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nameServers
Enclosing Method: AsyncDnsResolver()
File: src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala:85
Taint Flags:

```
82 requestId
83 }
84
85 private val resolvers: List[ActorRef] = clientFactory(context, nameServers)
86
87 // only supports DnsProtocol, not the deprecated Dns protocol
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 85 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
88 // AsyncDnsManager converts between the protocols to support the deprecated protocol
```

src/main/scala/akka/io/dns/internal/AsyncDnsManager.scala, line 70 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: AsyncDnsManager()
File: src/main/scala/akka/io/dns/internal/AsyncDnsManager.scala:70
Taint Flags:

```
67 implicit val ec: ExecutionContextExecutor = context.dispatcher
68
69 val settings = new DnsSettings(system, resolverConfig)
70 implicit val timeout: Timeout = Timeout(settings.ResolveTimeout)
71
72 private val resolver = {
73 val props: Props = FromConfig.props(
```

src/main/scala/akka/io/dns/internal/AsyncDnsManager.scala, line 74 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: AsyncDnsManager()
File: src/main/scala/akka/io/dns/internal/AsyncDnsManager.scala:74
Taint Flags:

```
71
72 private val resolver = {
73 val props: Props = FromConfig.props(
74 Props(provider.actorClass, settings, cache, (factory: ActorRefFactory, dns: List[InetSocketAddress]) => {
75 dns.map(ns => factory.actorOf(Props(new DnsClient(ns))))
76 }).withDeploy(Deploy.local).withDispatcher(dispatcher))
77 context.actorOf(props, name)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.pattern

src/main/scala/akka/pattern/StatusReply.scala, line 67 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: success
Enclosing Method: StatusReply()
File: src/main/scala/akka/pattern/StatusReply.scala:67
Taint Flags:

```
64 /**
65 * Scala API: A general purpose message for using as an Ack
66 */
67 val Ack: StatusReply[Done] = success(Done)
68
69 /**
70 * Java API: A general purpose message for using as an Ack
```

src/main/scala/akka/pattern/CircuitBreaker.scala, line 253 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akka$pattern$CircuitBreaker$$Closed
Enclosing Method: CircuitBreaker()
File: src/main/scala/akka/pattern/CircuitBreaker.scala:253
Taint Flags:

```
250 * Holds reference to current state of CircuitBreaker - *access only via helper methods*
251 */
252 @volatile
253 private[this] var _currentStateDoNotCallMeDirectly: State = Closed
254
255 /**
256 * Holds reference to current resetTimeout of CircuitBreaker - *access only via helper methods*
```

Package: akka.routing

src/main/scala/akka/routing/Balancing.scala, line 80 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/Balancing.scala, line 80 (Code Correctness: Constructor Invokes Overridable Function) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: BalancingPool()
File: src/main/scala/akka/routing/Balancing.scala:80
Taint Flags:

```
77 extends Pool {
78
79 def this(config: Config) =
80 this(nrOfInstances = config.getInt("nr-of-instances"))
81
82 /**
83 * Java API
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 180 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: checkParamAsProbability
Enclosing Method: DefaultOptimalSizeExploringResizer()
File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:180
Taint Flags:

```
177 numOfAdjacentSizesToConsiderDuringOptimization,
178 2,
179 "numOfAdjacentSizesToConsiderDuringOptimization")
180 checkParamAsProbability(chanceOfScalingDownWhenFull, "chanceOfScalingDownWhenFull")
181 checkParamAsPositiveNum(
182 numOfAdjacentSizesToConsiderDuringOptimization,
183 "numOfAdjacentSizesToConsiderDuringOptimization")
```

src/main/scala/akka/routing/Broadcast.scala, line 140 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/Broadcast.scala, line 140 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink: FunctionCall: $default$2
Enclosing Method: BroadcastGroup()
File: src/main/scala/akka/routing/Broadcast.scala:140
Taint Flags:

```
137 * @param routeePaths string representation of the actor paths of the routees, messages are
138 * sent with [[akka.actor.ActorSelection]] to these paths
139 */
140 def this(routeePaths: java.lang.Iterable[String]) = this(paths = immutableSeq(routeePaths))
141
142 override def paths(system: ActorSystem): immutable.Iterable[String] = this.paths
143
```

src/main/scala/akka/routing/Resizer.scala, line 142 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: DefaultResizer()
File: src/main/scala/akka/routing/Resizer.scala:142
Taint Flags:

```
139 /**
140 * Java API constructor for default values except bounds.
141 */
142 def this(lower: Int, upper: Int) = this(lowerBound = lower, upperBound = upper)
143
144 if (lowerBound < 0) throw new IllegalArgumentException("lowerBound must be >= 0, was: [%s]".format(lowerBound))
145 if (upperBound < 0) throw new IllegalArgumentException("upperBound must be >= 0, was: [%s]".format(upperBound))
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 391 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: ConsistentHashingGroup()
File: src/main/scala/akka/routing/ConsistentHashing.scala:391
Taint Flags:

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ConsistentHashing.scala, line 391 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
388 * @param routeePaths string representation of the actor paths of the routees, messages are
389 * sent with [[akka.actor.ActorSelection]] to these paths
390 */
391 def this(routeePaths: java.lang.Iterable[String]) = this(paths = immutableSeq(routeePaths))
392
393 override def paths(system: ActorSystem): immutable.Iterable[String] = this.paths
394
```

src/main/scala/akka/routing/RoundRobin.scala, line 144 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: RoundRobinGroup()
File: src/main/scala/akka/routing/RoundRobin.scala:144
Taint Flags:

```
141 extends Group {
142
143 def this(config: Config) =
144 this(paths = immutableSeq(config.getStringList("routees.paths")))
145
146 /**
147 * Java API
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 384 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: ConsistentHashingGroup()
File: src/main/scala/akka/routing/ConsistentHashing.scala:384
Taint Flags:

```
381 extends Group {
382
383 def this(config: Config) =
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ConsistentHashing.scala, line 384 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
384 this(paths = immutableSeq(config.getStringList("routees.paths")))
385
386 /**
387 * Java API
```

src/main/scala/akka/routing/Resizer.scala, line 142 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$7
Enclosing Method: DefaultResizer()
File: src/main/scala/akka/routing/Resizer.scala:142
Taint Flags:

```
139 /**
140 * Java API constructor for default values except bounds.
141 */
142 def this(lower: Int, upper: Int) = this(lowerBound = lower, upperBound = upper)
143
144 if (lowerBound < 0) throw new IllegalArgumentException("lowerBound must be >= 0, was: [%s]".format(lowerBound))
145 if (upperBound < 0) throw new IllegalArgumentException("upperBound must be >= 0, was: [%s]".format(upperBound))
```

src/main/scala/akka/routing/Random.scala, line 84 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: RandomPool()
File: src/main/scala/akka/routing/Random.scala:84
Taint Flags:

```
81 * Java API
82 * @param nr initial number of routees in the pool
83 */
84 def this(nr: Int) = this(nrOfInstances = nr)
85
86 override def createRouter(system: ActorSystem): Router = new Router(RandomRoutingLogic())
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.routing

src/main/scala/akka/routing/Random.scala, line 84 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
87
```

src/main/scala/akka/routing/TailChopping.scala, line 180 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$7
Enclosing Method: TailChoppingPool()
File: src/main/scala/akka/routing/TailChopping.scala:180
Taint Flags:

```
177 * @param interval duration after which next routee will be picked
178 */
179 def this(nr: Int, within: FiniteDuration, interval: FiniteDuration) =
180 this(nrOfInstances = nr, within = within, interval = interval)
181
182 /**
183 * Java API
```

src/main/scala/akka/routing/Broadcast.scala, line 133 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: BroadcastGroup()
File: src/main/scala/akka/routing/Broadcast.scala:133
Taint Flags:

```
130 extends Group {
131
132 def this(config: Config) =
133 this(paths = immutableSeq(config.getStringList("routees.paths")))
134
135 /**
136 * Java API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/SmallestMailbox.scala, line 196 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: SmallestMailboxPool()
File: src/main/scala/akka/routing/SmallestMailbox.scala:196
Taint Flags:

```
193 with PoolOverrideUnsetConfig[SmallestMailboxPool] {
194
195 def this(config: Config) =
196 this(
197 nrOfInstances = config.getInt("nr-of-instances"),
198 resizer = Resizer.fromConfig(config),
199 usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 391 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: ConsistentHashingGroup()
File: src/main/scala/akka/routing/ConsistentHashing.scala:391
Taint Flags:

```
388 * @param routeePaths string representation of the actor paths of the routees, messages are
389 * sent with [[akka.actor.ActorSelection]] to these paths
390 */
391 def this(routeePaths: java.lang.Iterable[String]) = this(paths = immutableSeq(routeePaths))
392
393 override def paths(system: ActorSystem): immutable.Iterable[String] = this.paths
394
```

src/main/scala/akka/routing/Resizer.scala, line 142 (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.routing

src/main/scala/akka/routing/Resizer.scala, line 142 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: DefaultResizer()
File: src/main/scala/akka/routing/Resizer.scala:142
Taint Flags:

```
139 /**
140 * Java API constructor for default values except bounds.
141 */
142 def this(lower: Int, upper: Int) = this(lowerBound = lower, upperBound = upper)
143
144 if (lowerBound < 0) throw new IllegalArgumentException("lowerBound must be >= 0, was: [%s]".format(lowerBound))
145 if (upperBound < 0) throw new IllegalArgumentException("upperBound must be >= 0, was: [%s]".format(upperBound))
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 299 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:299
Taint Flags:

```
296 with PoolOverrideUnsetConfig[ConsistentHashingPool] {
297
298 def this(config: Config) =
299 this(
300 nrOfInstances = config.getInt("nr-of-instances"),
301 resizer = Resizer.fromConfig(config),
302 usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 201 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: ScatterGatherFirstCompletedGroup()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 201 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala:201
Taint Flags:

```
198 * it will reply with [[akka.pattern.AskTimeoutException]] in a [[akka.actor.Status.Failure]]
199 */
200 def this(routeePaths: java.lang.Iterable[String], within: FiniteDuration) =
201 this(paths = immutableSeq(routeePaths), within = within)
202
203 /**
204 * Java API
```

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 127 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: ScatterGatherFirstCompletedPool()
File: src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala:127
Taint Flags:

```
124 * @param within expecting at least one reply within this duration, otherwise
125 * it will reply with [[akka.pattern.AskTimeoutException]] in a [[akka.actor.Status.Failure]]
126 */
127 def this(nr: Int, within: FiniteDuration) = this(nrOfInstances = nr, within = within)
128
129 /**
130 * Java API
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 185 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: checkParamAsPositiveNum
Enclosing Method: DefaultOptimalSizeExploringResizer()
File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:185
Taint Flags:

```
182 numOfAdjacentSizesToConsiderDuringOptimization,
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 185 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
183  "numOfAdjacentSizesToConsiderDuringOptimization")
184  checkParamAsPositiveNum(exploreStepSize, "exploreStepSize")
185  checkParamAsPositiveNum(downsizeRatio, "downsizeRatio")
186  checkParamAsProbability(explorationProbability, "explorationProbability")
187  checkParamAsProbability(weightOfLatestMetric, "weightOfLatestMetric")
188
```

src/main/scala/akka/routing/Resizer.scala, line 142 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: DefaultResizer()
File: src/main/scala/akka/routing/Resizer.scala:142
Taint Flags:

```
139  /**
140  * Java API constructor for default values except bounds.
141  */
142  def this(lower: Int, upper: Int) = this(lowerBound = lower, upperBound = upper)
143
144  if (lowerBound < 0) throw new IllegalArgumentException("lowerBound must be >= 0, was: [%s]".format(lowerBound))
145  if (upperBound < 0) throw new IllegalArgumentException("upperBound must be >= 0, was: [%s]".format(upperBound))
```

src/main/scala/akka/routing/RoundRobin.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: RoundRobinPool()
File: src/main/scala/akka/routing/RoundRobin.scala:92
Taint Flags:

```
89  * Java API
90  * @param nr initial number of routees in the pool
91  */
92  def this(nr: Int) = this(nrOfInstances = nr)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/RoundRobin.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
93
94  override def createRouter(system: ActorSystem): Router = new Router(RoundRobinRoutingLogic())
95
```

src/main/scala/akka/routing/Broadcast.scala, line 73 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: BroadcastPool()
File: src/main/scala/akka/routing/Broadcast.scala:73
Taint Flags:

```
70  with PoolOverrideUnsetConfig[BroadcastPool] {
71
72  def this(config: Config) =
73  this(
74  nrOfInstances = config.getInt("nr-of-instances"),
75  resizer = Resizer.fromConfig(config),
76  usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/Broadcast.scala, line 73 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: BroadcastPool()
File: src/main/scala/akka/routing/Broadcast.scala:73
Taint Flags:

```
70  with PoolOverrideUnsetConfig[BroadcastPool] {
71
72  def this(config: Config) =
73  this(
74  nrOfInstances = config.getInt("nr-of-instances"),
75  resizer = Resizer.fromConfig(config),
76  usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ConsistentHashing.scala, line 308 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:308
Taint Flags:

```
305 * Java API
306 * @param nr initial number of routees in the pool
307 */
308 def this(nr: Int) = this(nrOfInstances = nr)
309
310 override def createRouter(system: ActorSystem): Router =
311 new Router(ConsistentHashingRoutingLogic(system, virtualNodesFactor, hashMapping))
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 391 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: ConsistentHashingGroup()
File: src/main/scala/akka/routing/ConsistentHashing.scala:391
Taint Flags:

```
388 * @param routeePaths string representation of the actor paths of the routees, messages are
389 * sent with [[akka.actor.ActorSelection]] to these paths
390 */
391 def this(routeePaths: java.lang.Iterable[String]) = this(paths = immutableSeq(routeePaths))
392
393 override def paths(system: ActorSystem): immutable.Iterable[String] = this.paths
394
```

src/main/scala/akka/routing/Balancing.scala, line 86 (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.routing

src/main/scala/akka/routing/Balancing.scala, line 86 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: BalancingPool()
File: src/main/scala/akka/routing/Balancing.scala:86
Taint Flags:

```
83 * Java API
84 * @param nr initial number of routees in the pool
85 */
86 def this(nr: Int) = this(nrOfInstances = nr)
87
88 override def createRouter(system: ActorSystem): Router = new Router(BalancingRoutingLogic())
89
```

src/main/scala/akka/routing/TailChopping.scala, line 165 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: TailChoppingPool()
File: src/main/scala/akka/routing/TailChopping.scala:165
Taint Flags:

```
162 with PoolOverrideUnsetConfig[TailChoppingPool] {
163
164 def this(config: Config) =
165 this(
166 nrOfInstances = config.getInt("nr-of-instances"),
167 within = config.getMillisDuration("within"),
168 interval = config.getMillisDuration("tail-chopping-router.interval"),
```

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 115 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: ScatterGatherFirstCompletedPool()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 115 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala:115
Taint Flags:

```
112 with PoolOverrideUnsetConfig[ScatterGatherFirstCompletedPool] {
113
114 def this(config: Config) =
115 this(
116 nrOfInstances = config.getInt("nr-of-instances"),
117 within = config.getMillisDuration("within"),
118 resizer = Resizer.fromConfig(config),
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 186 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: checkParamAsProbability
Enclosing Method: DefaultOptimalSizeExploringResizer()
File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:186
Taint Flags:

```
183 "numOfAdjacentSizesToConsiderDuringOptimization")
184 checkParamAsPositiveNum(exploreStepSize, "exploreStepSize")
185 checkParamAsPositiveNum(downsizeRatio, "downsizeRatio")
186 checkParamAsProbability(explorationProbability, "explorationProbability")
187 checkParamAsProbability(weightOfLatestMetric, "weightOfLatestMetric")
188
189 private val actionInternalNanos = actionInterval.toNanos
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 184 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: checkParamAsPositiveNum
Enclosing Method: DefaultOptimalSizeExploringResizer()
File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:184
Taint Flags:

```
181 checkParamAsPositiveNum(
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 184 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
182 numOfAdjacentSizesToConsiderDuringOptimization,
183 "numOfAdjacentSizesToConsiderDuringOptimization")
184 checkParamAsPositiveNum(exploreStepSize, "exploreStepSize")
185 checkParamAsPositiveNum(downsizeRatio, "downsizeRatio")
186 checkParamAsProbability(explorationProbability, "explorationProbability")
187 checkParamAsProbability(weightOfLatestMetric, "weightOfLatestMetric")
```

src/main/scala/akka/routing/SmallestMailbox.scala, line 205 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: SmallestMailboxPool()
File: src/main/scala/akka/routing/SmallestMailbox.scala:205
Taint Flags:

```
202 * Java API
203 * @param nr initial number of routees in the pool
204 */
205 def this(nr: Int) = this(nrOfInstances = nr)
206
207 override def createRouter(system: ActorSystem): Router = new Router(SmallestMailboxRoutingLogic())
208
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 384 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: ConsistentHashingGroup()
File: src/main/scala/akka/routing/ConsistentHashing.scala:384
Taint Flags:

```
381 extends Group {
382
383 def this(config: Config) =
384 this(paths = immutableSeq(config.getStringList("routees.paths")))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ConsistentHashing.scala, line 384 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
385
386 /**
387 * Java API
```

src/main/scala/akka/routing/MurmurHash.scala, line 59 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akka$routing$MurmurHash$$hiddenMagicB
Enclosing Method: MurmurHash()
File: src/main/scala/akka/routing/MurmurHash.scala:59
Taint Flags:

```
56 Iterator.iterate(hiddenMagicA)(nextMagicA).take(23).toArray
57
58 /** The first 23 magic integers from the second stream are stored here */
59 private val storedMagicB: Array[Int] =
60 Iterator.iterate(hiddenMagicB)(nextMagicB).take(23).toArray
61
62 /** Begin a new hash with a seed value. */
```

src/main/scala/akka/routing/RoutedActorCell.scala, line 156 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: cell
Enclosing Method: RouterActor()
File: src/main/scala/akka/routing/RoutedActorCell.scala:156
Taint Flags:

```
153 throw ActorInitializationException("Router actor can only be used in RoutedActorRef, not in " + context.getClass)
154 }
155
156 val routingLogicController: Option[ActorRef] = cell.routerConfig
157 .routingLogicController(cell.router.logic)
158 .map(props => context.actorOf(props.withDispatcher(context.props.dispatcher), name = "routingLogicController"))
159
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/RoutedActorCell.scala, line 156 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: cell
Enclosing Method: RouterActor()
File: src/main/scala/akka/routing/RoutedActorCell.scala:156
Taint Flags:

```
153 throw ActorInitializationException("Router actor can only be used in RoutedActorRef, not in " + context.getClass)
154 }
155
156 val routingLogicController: Option[ActorRef] = cell.routerConfig
157 .routingLogicController(cell.router.logic)
158 .map(props => context.actorOf(props.withDispatcher(context.props.dispatcher), name = "routingLogicController"))
159
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 299 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:299
Taint Flags:

```
296 with PoolOverrideUnsetConfig[ConsistentHashingPool] {
297
298 def this(config: Config) =
299 this(
300 nrOfInstances = config.getInt("nr-of-instances"),
301 resizer = Resizer.fromConfig(config),
302 usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 115 (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.routing

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 115 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: ScatterGatherFirstCompletedPool()
File: src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala:115
Taint Flags:

```
112 with PoolOverrideUnsetConfig[ScatterGatherFirstCompletedPool] {
113
114 def this(config: Config) =
115 this(
116 nrOfInstances = config.getInt("nr-of-instances"),
117 within = config.getMillisDuration("within"),
118 resizer = Resizer.fromConfig(config),
```

src/main/scala/akka/routing/Balancing.scala, line 80 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: BalancingPool()
File: src/main/scala/akka/routing/Balancing.scala:80
Taint Flags:

```
77 extends Pool {
78
79 def this(config: Config) =
80 this(nrOfInstances = config.getInt("nr-of-instances"))
81
82 /**
83 * Java API
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 187 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: checkParamAsProbability
Enclosing Method: DefaultOptimalSizeExploringResizer()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 187 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:187
Taint Flags:

```
184 checkParamAsPositiveNum(exploreStepSize, "exploreStepSize")
185 checkParamAsPositiveNum(downsizeRatio, "downsizeRatio")
186 checkParamAsProbability(explorationProbability, "explorationProbability")
187 checkParamAsProbability(weightOfLatestMetric, "weightOfLatestMetric")
188
189 private val actionInternalNanos = actionInterval.toNanos
190
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 308 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:308
Taint Flags:

```
305 * Java API
306 * @param nr initial number of routees in the pool
307 */
308 def this(nr: Int) = this(nrOfInstances = nr)
309
310 override def createRouter(system: ActorSystem): Router =
311 new Router(ConsistentHashingRoutingLogic(system, virtualNodesFactor, hashMapping))
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 308 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$7
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:308
Taint Flags:

```
305 * Java API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ConsistentHashing.scala, line 308 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
306 * @param nr initial number of routees in the pool
307 */
308 def this(nr: Int) = this(nrOfInstances = nr)
309
310 override def createRouter(system: ActorSystem): Router =
311 new Router(ConsistentHashingRoutingLogic(system, virtualNodesFactor, hashMapping))
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 384 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: ConsistentHashingGroup()
File: src/main/scala/akka/routing/ConsistentHashing.scala:384
Taint Flags:

```
381 extends Group {
382
383 def this(config: Config) =
384 this(paths = immutableSeq(config.getStringList("routees.paths")))
385
386 /**
387 * Java API
```

src/main/scala/akka/routing/Random.scala, line 142 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: RandomGroup()
File: src/main/scala/akka/routing/Random.scala:142
Taint Flags:

```
139 * @param routeePaths string representation of the actor paths of the routees, messages are
140 * sent with [[akka.actor.ActorSelection]] to these paths
141 */
142 def this(routeePaths: java.lang.Iterable[String]) = this(paths = immutableSeq(routeePaths))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/Random.scala, line 142 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
143
144 override def paths(system: ActorSystem): immutable.Iterable[String] = this.paths
145
```

src/main/scala/akka/routing/Balancing.scala, line 86 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: BalancingPool()
File: src/main/scala/akka/routing/Balancing.scala:86
Taint Flags:

```
83 * Java API
84 * @param nr initial number of routees in the pool
85 */
86 def this(nr: Int) = this(nrOfInstances = nr)
87
88 override def createRouter(system: ActorSystem): Router = new Router(BalancingRoutingLogic())
89
```

src/main/scala/akka/routing/TailChopping.scala, line 273 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: TailChoppingGroup()
File: src/main/scala/akka/routing/TailChopping.scala:273
Taint Flags:

```
270 * @param interval duration after which next routee will be picked
271 */
272 def this(routeePaths: java.lang.Iterable[String], within: FiniteDuration, interval: FiniteDuration) =
273 this(paths = immutableSeq(routeePaths), within = within, interval = interval)
274
275 /**
276 * Java API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 176 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: checkParamLowerBound
Enclosing Method: DefaultOptimalSizeExploringResizer()
File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:176
Taint Flags:

```
173 throw new IllegalArgumentException(
174 "upperBound must be >= lowerBound, was: [%s] < [%s]".format(upperBound, lowerBound))
175
176 checkParamLowerBound(
177 numOfAdjacentSizesToConsiderDuringOptimization,
178 2,
179 "numOfAdjacentSizesToConsiderDuringOptimization")
```

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 211 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: ScatterGatherFirstCompletedGroup()
File: src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala:211
Taint Flags:

```
208 * it will reply with [[akka.pattern.AskTimeoutException]] in a [[akka.actor.Status.Failure]]
209 */
210 def this(routeePaths: java.lang.Iterable[String], within: java.time.Duration) =
211 this(immutableSeq(routeePaths), within.asScala)
212
213 override def paths(system: ActorSystem): immutable.Iterable[String] = this.paths
214
```

src/main/scala/akka/routing/RoundRobin.scala, line 83 (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.routing

src/main/scala/akka/routing/RoundRobin.scala, line 83 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: RoundRobinPool()
File: src/main/scala/akka/routing/RoundRobin.scala:83
Taint Flags:

```
80 with PoolOverrideUnsetConfig[RoundRobinPool] {
81
82 def this(config: Config) =
83 this(
84 nrOfInstances = config.getInt("nr-of-instances"),
85 resizer = Resizer.fromConfig(config),
86 usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/SmallestMailbox.scala, line 196 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: SmallestMailboxPool()
File: src/main/scala/akka/routing/SmallestMailbox.scala:196
Taint Flags:

```
193 with PoolOverrideUnsetConfig[SmallestMailboxPool] {
194
195 def this(config: Config) =
196 this(
197 nrOfInstances = config.getInt("nr-of-instances"),
198 resizer = Resizer.fromConfig(config),
199 usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 170 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: checkParamAsPositiveNum
Enclosing Method: DefaultOptimalSizeExploringResizer()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 170 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:170
Taint Flags:

```
167 if (value < lowerBound)
168 throw new IllegalArgumentException(s"$paramName must be >= $lowerBound, was: [%s]".format(value))
169
170 checkParamAsPositiveNum(lowerBound, "lowerBound")
171 checkParamAsPositiveNum(upperBound, "upperBound")
172 if (upperBound < lowerBound)
173 throw new IllegalArgumentException(
```

src/main/scala/akka/routing/RoundRobin.scala, line 151 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: RoundRobinGroup()
File: src/main/scala/akka/routing/RoundRobin.scala:151
Taint Flags:

```
148 * @param routeePaths string representation of the actor paths of the routees, messages are
149 * sent with [[akka.actor.ActorSelection]] to these paths
150 */
151 def this(routeePaths: java.lang.Iterable[String]) = this(paths = immutableSeq(routeePaths))
152
153 override def paths(system: ActorSystem): immutable.Iterable[String] = this.paths
154
```

src/main/scala/akka/routing/Random.scala, line 84 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: RandomPool()
File: src/main/scala/akka/routing/Random.scala:84
Taint Flags:

```
81 * Java API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/Random.scala, line 84 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
82  * @param nr initial number of routees in the pool
83  */
84  def this(nr: Int) = this(nrOfInstances = nr)
85
86  override def createRouter(system: ActorSystem): Router = new Router(RandomRoutingLogic())
87
```

src/main/scala/akka/routing/TailChopping.scala, line 180 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: TailChoppingPool()
File: src/main/scala/akka/routing/TailChopping.scala:180
Taint Flags:

```
177  * @param interval duration after which next routee will be picked
178  */
179  def this(nr: Int, within: FiniteDuration, interval: FiniteDuration) =
180  this(nrOfInstances = nr, within = within, interval = interval)
181
182  /**
183  * Java API
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 308 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:308
Taint Flags:

```
305  * Java API
306  * @param nr initial number of routees in the pool
307  */
308  def this(nr: Int) = this(nrOfInstances = nr)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ConsistentHashing.scala, line 308 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
309
310 override def createRouter(system: ActorSystem): Router =
311 new Router(ConsistentHashingRoutingLogic(system, virtualNodesFactor, hashMapping))
```

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 127 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: ScatterGatherFirstCompletedPool()
File: src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala:127
Taint Flags:

```
124 * @param within expecting at least one reply within this duration, otherwise
125 * it will reply with [[akka.pattern.AskTimeoutException]] in a [[akka.actor.Status.Failure]]
126 */
127 def this(nr: Int, within: FiniteDuration) = this(nrOfInstances = nr, within = within)
128
129 /**
130 * Java API
```

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 127 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: ScatterGatherFirstCompletedPool()
File: src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala:127
Taint Flags:

```
124 * @param within expecting at least one reply within this duration, otherwise
125 * it will reply with [[akka.pattern.AskTimeoutException]] in a [[akka.actor.Status.Failure]]
126 */
127 def this(nr: Int, within: FiniteDuration) = this(nrOfInstances = nr, within = within)
128
129 /**
130 * Java API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/TailChopping.scala, line 165 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: TailChoppingPool()
File: src/main/scala/akka/routing/TailChopping.scala:165
Taint Flags:

```
162 with PoolOverrideUnsetConfig[TailChoppingPool] {
163
164 def this(config: Config) =
165 this(
166 nrOfInstances = config.getInt("nr-of-instances"),
167 within = config.getMillisDuration("within"),
168 interval = config.getMillisDuration("tail-chopping-router.interval"),
```

src/main/scala/akka/routing/TailChopping.scala, line 180 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$6
Enclosing Method: TailChoppingPool()
File: src/main/scala/akka/routing/TailChopping.scala:180
Taint Flags:

```
177 * @param interval duration after which next routee will be picked
178 */
179 def this(nr: Int, within: FiniteDuration, interval: FiniteDuration) =
180 this(nrOfInstances = nr, within = within, interval = interval)
181
182 /**
183 * Java API
```

src/main/scala/akka/routing/Broadcast.scala, line 82 (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.routing

src/main/scala/akka/routing/Broadcast.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: BroadcastPool()
File: src/main/scala/akka/routing/Broadcast.scala:82
Taint Flags:

```
79 * Java API
80 * @param nr initial number of routees in the pool
81 */
82 def this(nr: Int) = this(nrOfInstances = nr)
83
84 override def createRouter(system: ActorSystem): Router = new Router(BroadcastRoutingLogic())
85
```

src/main/scala/akka/routing/Broadcast.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: BroadcastPool()
File: src/main/scala/akka/routing/Broadcast.scala:82
Taint Flags:

```
79 * Java API
80 * @param nr initial number of routees in the pool
81 */
82 def this(nr: Int) = this(nrOfInstances = nr)
83
84 override def createRouter(system: ActorSystem): Router = new Router(BroadcastRoutingLogic())
85
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 181 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: checkParamAsPositiveNum
Enclosing Method: DefaultOptimalSizeExploringResizer()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 181 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:181
Taint Flags:

```
178  2,
179  "numOfAdjacentSizesToConsiderDuringOptimization")
180  checkParamAsProbability(chanceOfScalingDownWhenFull, "chanceOfScalingDownWhenFull")
181  checkParamAsPositiveNum(
182  numOfAdjacentSizesToConsiderDuringOptimization,
183  "numOfAdjacentSizesToConsiderDuringOptimization")
184  checkParamAsPositiveNum(exploreStepSize, "exploreStepSize")
```

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 127 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: ScatterGatherFirstCompletedPool()
File: src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala:127
Taint Flags:

```
124  * @param within expecting at least one reply within this duration, otherwise
125  * it will reply with [[akka.pattern.AskTimeoutException]] in a [[akka.actor.Status.Failure]]
126  */
127  def this(nr: Int, within: FiniteDuration) = this(nrOfInstances = nr, within = within)
128
129  /**
130  * Java API
```

src/main/scala/akka/routing/TailChopping.scala, line 180 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: TailChoppingPool()
File: src/main/scala/akka/routing/TailChopping.scala:180
Taint Flags:

```
177  * @param interval duration after which next routee will be picked
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/TailChopping.scala, line 180 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
178 */
179 def this(nr: Int, within: FiniteDuration, interval: FiniteDuration) =
180 this(nrOfInstances = nr, within = within, interval = interval)
181
182 /**
183 * Java API
```

src/main/scala/akka/routing/SmallestMailbox.scala, line 205 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: SmallestMailboxPool()
File: src/main/scala/akka/routing/SmallestMailbox.scala:205
Taint Flags:

```
202 * Java API
203 * @param nr initial number of routees in the pool
204 */
205 def this(nr: Int) = this(nrOfInstances = nr)
206
207 override def createRouter(system: ActorSystem): Router = new Router(SmallestMailboxRoutingLogic())
208
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 299 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:299
Taint Flags:

```
296 with PoolOverrideUnsetConfig[ConsistentHashingPool] {
297
298 def this(config: Config) =
299 this(
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ConsistentHashing.scala, line 299 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
300  nrOfInstances = config.getInt("nr-of-instances"),
301  resizer = Resizer.fromConfig(config),
302  usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/Random.scala, line 84 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: RandomPool()
File: src/main/scala/akka/routing/Random.scala:84
Taint Flags:

```
81  * Java API
82  * @param nr initial number of routees in the pool
83  */
84  def this(nr: Int) = this(nrOfInstances = nr)
85
86  override def createRouter(system: ActorSystem): Router = new Router(RandomRoutingLogic())
87
```

src/main/scala/akka/routing/Broadcast.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: BroadcastPool()
File: src/main/scala/akka/routing/Broadcast.scala:82
Taint Flags:

```
79  * Java API
80  * @param nr initial number of routees in the pool
81  */
82  def this(nr: Int) = this(nrOfInstances = nr)
83
84  override def createRouter(system: ActorSystem): Router = new Router(BroadcastRoutingLogic())
85
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/TailChopping.scala, line 284 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: TailChoppingGroup()
File: src/main/scala/akka/routing/TailChopping.scala:284
Taint Flags:

```
281 * @param interval duration after which next routee will be picked
282 */
283 def this(routeePaths: java.lang.Iterable[String], within: java.time.Duration, interval: java.time.Duration) =
284 this(immutableSeq(routeePaths), within.asScala, interval.asScala)
285
286 override def createRouter(system: ActorSystem): Router =
287 new Router(
```

src/main/scala/akka/routing/MurmurHash.scala, line 55 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akka$routing$MurmurHash$$hiddenMagicA
Enclosing Method: MurmurHash()
File: src/main/scala/akka/routing/MurmurHash.scala:55
Taint Flags:

```
52 final private val seedArray: Int = 0x3c074a61
53
54 /** The first 23 magic integers from the first stream are stored here */
55 private val storedMagicA: Array[Int] =
56 Iterator.iterate(hiddenMagicA)(nextMagicA).take(23).toArray
57
58 /** The first 23 magic integers from the second stream are stored here */
```

src/main/scala/akka/routing/Random.scala, line 135 (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.routing

src/main/scala/akka/routing/Random.scala, line 135 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: RandomGroup()
File: src/main/scala/akka/routing/Random.scala:135
Taint Flags:

```
132 extends Group {
133
134 def this(config: Config) =
135 this(paths = immutableSeq(config.getStringList("routees.paths")))
136
137 /**
138 * Java API
```

src/main/scala/akka/routing/Broadcast.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: BroadcastPool()
File: src/main/scala/akka/routing/Broadcast.scala:82
Taint Flags:

```
79 * Java API
80 * @param nr initial number of routees in the pool
81 */
82 def this(nr: Int) = this(nrOfInstances = nr)
83
84 override def createRouter(system: ActorSystem): Router = new Router(BroadcastRoutingLogic())
85
```

src/main/scala/akka/routing/RoundRobin.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: RoundRobinPool()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/RoundRobin.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/routing/RoundRobin.scala:92
Taint Flags:

```
89 * Java API
90 * @param nr initial number of routees in the pool
91 */
92 def this(nr: Int) = this(nrOfInstances = nr)
93
94 override def createRouter(system: ActorSystem): Router = new Router(RoundRobinRoutingLogic())
95
```

src/main/scala/akka/routing/Random.scala, line 75 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: RandomPool()
File: src/main/scala/akka/routing/Random.scala:75
Taint Flags:

```
72 with PoolOverrideUnsetConfig[RandomPool] {
73
74 def this(config: Config) =
75 this(
76 nrOfInstances = config.getInt("nr-of-instances"),
77 resizer = Resizer.fromConfig(config),
78 usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/Resizer.scala, line 142 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: DefaultResizer()
File: src/main/scala/akka/routing/Resizer.scala:142
Taint Flags:

```
139 /**
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/Resizer.scala, line 142 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
140  * Java API constructor for default values except bounds.
141  */
142  def this(lower: Int, upper: Int) = this(lowerBound = lower, upperBound = upper)
143
144  if (lowerBound < 0) throw new IllegalArgumentException("lowerBound must be >= 0, was: [%s]".format(lowerBound))
145  if (upperBound < 0) throw new IllegalArgumentException("upperBound must be >= 0, was: [%s]".format(upperBound))
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 308 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:308
Taint Flags:

```
305  * Java API
306  * @param nr initial number of routees in the pool
307  */
308  def this(nr: Int) = this(nrOfInstances = nr)
309
310  override def createRouter(system: ActorSystem): Router =
311  new Router(ConsistentHashingRoutingLogic(system, virtualNodesFactor, hashMapping))
```

src/main/scala/akka/routing/RoundRobin.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: RoundRobinPool()
File: src/main/scala/akka/routing/RoundRobin.scala:92
Taint Flags:

```
89  * Java API
90  * @param nr initial number of routees in the pool
91  */
92  def this(nr: Int) = this(nrOfInstances = nr)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/RoundRobin.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
93
94  override def createRouter(system: ActorSystem): Router = new Router(RoundRobinRoutingLogic())
95
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 308 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:308
Taint Flags:

```
305  * Java API
306  * @param nr initial number of routees in the pool
307  */
308  def this(nr: Int) = this(nrOfInstances = nr)
309
310  override def createRouter(system: ActorSystem): Router =
311  new Router(ConsistentHashingRoutingLogic(system, virtualNodesFactor, hashMapping))
```

src/main/scala/akka/routing/ConsistentHashing.scala, line 299 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: ConsistentHashingPool()
File: src/main/scala/akka/routing/ConsistentHashing.scala:299
Taint Flags:

```
296  with PoolOverrideUnsetConfig[ConsistentHashingPool] {
297
298  def this(config: Config) =
299  this(
300  nrOfInstances = config.getInt("nr-of-instances"),
301  resizer = Resizer.fromConfig(config),
302  usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/SmallestMailbox.scala, line 205 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: SmallestMailboxPool()
File: src/main/scala/akka/routing/SmallestMailbox.scala:205
Taint Flags:

```
202 * Java API
203 * @param nr initial number of routees in the pool
204 */
205 def this(nr: Int) = this(nrOfInstances = nr)
206
207 override def createRouter(system: ActorSystem): Router = new Router(SmallestMailboxRoutingLogic())
208
```

src/main/scala/akka/routing/Random.scala, line 84 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: RandomPool()
File: src/main/scala/akka/routing/Random.scala:84
Taint Flags:

```
81 * Java API
82 * @param nr initial number of routees in the pool
83 */
84 def this(nr: Int) = this(nrOfInstances = nr)
85
86 override def createRouter(system: ActorSystem): Router = new Router(RandomRoutingLogic())
87
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 171 (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.routing

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 171 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: checkParamAsPositiveNum
Enclosing Method: DefaultOptimalSizeExploringResizer()
File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:171
Taint Flags:

```
168 throw new IllegalArgumentException(s"$paramName must be >= $lowerBound, was: [%s]".format(value))
169
170 checkParamAsPositiveNum(lowerBound, "lowerBound")
171 checkParamAsPositiveNum(upperBound, "upperBound")
172 if (upperBound < lowerBound)
173 throw new IllegalArgumentException(
174 "upperBound must be >= lowerBound, was: [%s] < [%s]".format(upperBound, lowerBound))
```

src/main/scala/akka/routing/Random.scala, line 75 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: RandomPool()
File: src/main/scala/akka/routing/Random.scala:75
Taint Flags:

```
72 with PoolOverrideUnsetConfig[RandomPool] {
73
74 def this(config: Config) =
75 this(
76 nrOfInstances = config.getInt("nr-of-instances"),
77 resizer = Resizer.fromConfig(config),
78 usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/RoundRobin.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$5
Enclosing Method: RoundRobinPool()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/RoundRobin.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/routing/RoundRobin.scala:92
Taint Flags:

```
89 * Java API
90 * @param nr initial number of routees in the pool
91 */
92 def this(nr: Int) = this(nrOfInstances = nr)
93
94 override def createRouter(system: ActorSystem): Router = new Router(RoundRobinRoutingLogic())
95
```

src/main/scala/akka/routing/TailChopping.scala, line 259 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$4
Enclosing Method: TailChoppingGroup()
File: src/main/scala/akka/routing/TailChopping.scala:259
Taint Flags:

```
256 extends Group {
257
258 def this(config: Config) =
259 this(
260 paths = immutableSeq(config.getStringList("routees.paths")),
261 within = config.getMillisDuration("within"),
262 interval = config.getMillisDuration("tail-chopping-router.interval"))
```

src/main/scala/akka/routing/SmallestMailbox.scala, line 205 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: SmallestMailboxPool()
File: src/main/scala/akka/routing/SmallestMailbox.scala:205
Taint Flags:

```
202 * Java API
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/SmallestMailbox.scala, line 205 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
203  * @param nr initial number of routees in the pool
204  */
205  def this(nr: Int) = this(nrOfInstances = nr)
206
207  override def createRouter(system: ActorSystem): Router = new Router(SmallestMailboxRoutingLogic())
208
```

src/main/scala/akka/routing/RoundRobin.scala, line 83 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: RoundRobinPool()
File: src/main/scala/akka/routing/RoundRobin.scala:83
Taint Flags:

```
80  with PoolOverrideUnsetConfig[RoundRobinPool] {
81
82  def this(config: Config) =
83  this(
84  nrOfInstances = config.getInt("nr-of-instances"),
85  resizer = Resizer.fromConfig(config),
86  usePoolDispatcher = config.hasPath("pool-dispatcher"))
```

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 191 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: ScatterGatherFirstCompletedGroup()
File: src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala:191
Taint Flags:

```
188  extends Group {
189
190  def this(config: Config) =
191  this(paths = immutableSeq(config.getStringList("routees.paths")), within = config.getMillisDuration("within"))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.routing

src/main/scala/akka/routing/ScatterGatherFirstCompleted.scala, line 191 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
192
193 /**
194 * Java API
```

Package: akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 426 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: ensureOnlyAllowedSerializers
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:426
Taint Flags:

```
423 private val serializers: Map[String, Serializer] = {
424 val fromConfig = for ((k: String, v: String) <- settings.Serializers) yield k -> serializerOf(k, v).get
425 val result = fromConfig ++ serializerDetails.map(d => d.alias -> d.serializer)
426 ensureOnlyAllowedSerializers(result.iterator.map { case (_, ser) => ser })
427 result
428 }
429
```

src/main/scala/akka/serialization/Serialization.scala, line 445 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: ensureOnlyAllowedSerializers
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:445
Taint Flags:

```
442 }
443
444 val result = sort(fromConfig ++ fromSettings)
445 ensureOnlyAllowedSerializers(result.iterator.map { case (_, ser) => ser })
446 result.foreach { case (clazz, ser) => warnUnexpectedNonAkkaSerializer(clazz, ser) }
447 result
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 445 (Code Correctness: Constructor Invokes Overridable Function) **Low**

```
448 }
```

src/main/scala/akka/serialization/Serialization.scala, line 528 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serializerByIdentity
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:528
Taint Flags:

```
525 private val quickSerializerByIdentity: Array[Serializer] = {
526 val size = 1024
527 val table = new Array[Serializer](size)
528 serializerByIdentity.foreach {
529 case (id, ser) => if (0 <= id && id < size) table(id) = ser
530 }
531 table
```

src/main/scala/akka/serialization/Serialization.scala, line 444 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: sort
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:444
Taint Flags:

```
441 detail.useFor.map(clazz => clazz -> detail.serializer)
442 }
443
444 val result = sort(fromConfig ++ fromSettings)
445 ensureOnlyAllowedSerializers(result.iterator.map { case (_, ser) => ser })
446 result.foreach { case (clazz, ser) => warnUnexpectedNonAkkaSerializer(clazz, ser) }
447 result
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 500 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bindings
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:500
Taint Flags:

```
497 * serializerMap is a Map whose keys is the class that is serializable and values is the serializer
498 * to be used for that class.
499 */
500 private val serializerMap: ConcurrentHashMap[Class[_], Serializer] =
501 bindings.foldLeft(new ConcurrentHashMap[Class[_], Serializer]) { case (map, (c, s)) => map.put(c, s); map }
502
503 /**
```

src/main/scala/akka/serialization/Serialization.scala, line 506 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serializers
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:506
Taint Flags:

```
503 /**
504 * Maps from a Serializer Identity (Int) to a Serializer instance (optimization)
505 */
506 val serializerByIdentity: Map[Int, Serializer] = {
507 val zero: Map[Int, Serializer] = Map(NullSerializer.identifier -> NullSerializer)
508 serializers.foldLeft(zero) {
509 case (acc, (_, ser)) =>
```

src/main/scala/akka/serialization/Serialization.scala, line 425 (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.serialization

src/main/scala/akka/serialization/Serialization.scala, line 425 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: serializerDetails
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:425
Taint Flags:

```
422 */
423 private val serializers: Map[String, Serializer] = {
424 val fromConfig = for ((k: String, v: String) <- settings.Serializers) yield k -> serializerOf(k, v).get
425 val result = fromConfig ++ serializerDetails.map(d => d.alias -> d.serializer)
426 ensureOnlyAllowedSerializers(result.iterator.map { case (_, ser) => ser })
427 result
428 }
```

src/main/scala/akka/serialization/Serialization.scala, line 440 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serializerDetails
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:440
Taint Flags:

```
437 if alias != "none" && checkGoogleProtobuf(className) && checkAkkaProtobuf(className)
438 } yield (system.dynamicAccess.getClassFor[Any](className).get, serializers(alias))
439
440 val fromSettings = serializerDetails.flatMap { detail =>
441 detail.useFor.map(clazz => clazz -> detail.serializer)
442 }
443
```

src/main/scala/akka/serialization/Serialization.scala, line 424 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: Serialization()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 424 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/serialization/Serialization.scala:424
Taint Flags:

```
421 * By default always contains the following mapping: "java" -> akka.serialization.JavaSerializer
422 */
423 private val serializers: Map[String, Serializer] = {
424 val fromConfig = for ((k: String, v: String) <- settings.Serializers) yield k -> serializerOf(k, v).get
425 val result = fromConfig ++ serializerDetails.map(d => d.alias -> d.serializer)
426 ensureOnlyAllowedSerializers(result.iterator.map { case (_, ser) => ser })
427 result
```

src/main/scala/akka/serialization/Serialization.scala, line 435 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:435
Taint Flags:

```
432 * It is primarily ordered by the most specific classes first, and secondly in the configured order.
433 */
434 private[akka] val bindings: immutable.Seq[ClassSerializer] = {
435 val fromConfig = for {
436 (className: String, alias: String) <- settings.SerializationBindings
437 if alias != "none" && checkGoogleProtobuf(className) && checkAkkaProtobuf(className)
438 } yield (system.dynamicAccess.getClassFor[Any](className).get, serializers(alias))
```

src/main/scala/akka/serialization/Serialization.scala, line 547 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:547
Taint Flags:

```
544 serializerByIdentity(id)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 547 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
545 }
546
547 private val isJavaSerializationWarningEnabled =
548 settings.config.getBoolean("akka.actor.warn-about-java-serializer-usage")
549 private val isWarningOnNoVerificationEnabled =
550 settings.config.getBoolean("akka.actor.warn-on-no-serialization-verification")
```

src/main/scala/akka/serialization/Serialization.scala, line 549 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: Serialization()
File: src/main/scala/akka/serialization/Serialization.scala:549
Taint Flags:

```
546
547 private val isJavaSerializationWarningEnabled =
548 settings.config.getBoolean("akka.actor.warn-about-java-serializer-usage")
549 private val isWarningOnNoVerificationEnabled =
550 settings.config.getBoolean("akka.actor.warn-on-no-serialization-verification")
551
552 private def isDisallowedJavaSerializer(serializer: Serializer): Boolean = {
```

Package: akka.util

src/main/scala/akka/util/PrettyByteString.scala, line 13 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: indentDepth
Enclosing Method: PrettyByteString()
File: src/main/scala/akka/util/PrettyByteString.scala:13
Taint Flags:

```
10 */
11 private[akka] object PrettyByteString {
12  private val indentDepth = 2
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

src/main/scala/akka/util/PrettyByteString.scala, line 13 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
13 private val indent = " " * (indentDepth + 1)
14
15 implicit class asPretty(bs: ByteString) {
16 def prettyPrint(maxBytes: Int = 16 * 5): String = formatBytes(bs, maxBytes).mkString("\n")
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 35 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: createNotFullCondition
Enclosing Method: BoundedBlockingQueue()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:35
Taint Flags:

```
32
33 protected val lock = createLock()
34 protected val notEmpty = createNotEmptyCondition()
35 protected val notFull = createNotFullCondition()
36
37 protected def createLock(): ReentrantLock = new ReentrantLock(false)
38 protected def createNotEmptyCondition(): Condition = lock.newCondition()
```

src/main/scala/akka/util/SegmentedRecencyList.scala, line 51 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: levels
Enclosing Method: SegmentedRecencyList()
File: src/main/scala/akka/util/SegmentedRecencyList.scala:51
Taint Flags:

```
48
49 private val levels = limits.size
50 private val lowest = 0
51 private val highest = levels - 1
52
53 private val segments = IndexedSeq.fill(levels)(
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.util

src/main/scala/akka/util/SegmentedRecencyList.scala, line 51 (Code Correctness: Constructor Invokes Overridable Function) **Low**

```
54 new DoubleLinkedList[Node[A]](
```

src/main/scala/akka/util/SegmentedRecencyList.scala, line 53 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: levels
Enclosing Method: SegmentedRecencyList()
File: src/main/scala/akka/util/SegmentedRecencyList.scala:53
Taint Flags:

```
50 private val lowest = 0
51 private val highest = levels - 1
52
53 private val segments = IndexedSeq.fill(levels)(
54 new DoubleLinkedList[Node[A]](
55 getPrevious = _.lessRecent,
56 getNext = _.moreRecent,
```

src/main/scala/akka/util/SegmentedRecencyList.scala, line 60 (Code Correctness: Constructor Invokes Overridable Function) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: levels
Enclosing Method: SegmentedRecencyList()
File: src/main/scala/akka/util/SegmentedRecencyList.scala:60
Taint Flags:

```
57 setPrevious = (node, previous) => node.lessRecent = previous,
58 setNext = (node, next) => node.moreRecent = next))
59
60 private val sizes = mutable.IndexedSeq.fill(levels)(0)
61
62 private val overallRecency = new DoubleLinkedList[Node[A]](
63 getPrevious = _.overallLessRecent,
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

src/main/scala/akka/util/Reflect.scala, line 34 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: liftedTree1
Enclosing Method: Reflect()
File: src/main/scala/akka/util/Reflect.scala:34
Taint Flags:

```
31 *
32 * Hint: when comparing to Thread.currentThread().getStackTrace, add two levels.
33 */
34 val getCallerClass: Option[Int => Class[_]] = {
35 try {
36 val c = Class.forName("sun.reflect.Reflection")
37 val m = c.getMethod("getCallerClass", Array(classOf[Int]): _*)
```

src/main/scala/akka/util/WildcardIndex.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$3
Enclosing Method: WildcardTree()
File: src/main/scala/akka/util/WildcardIndex.scala:41
Taint Flags:

```
38 }
39
40 private[akka] object WildcardTree {
41 private val empty = new WildcardTree[Nothing]()
42 def apply[T](): WildcardTree[T] = empty.asInstanceOf[WildcardTree[T]]
43 }
44
```

src/main/scala/akka/util/ManifestInfo.scala, line 116 (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.util

src/main/scala/akka/util/ManifestInfo.scala, line 116 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: liftedTree1
Enclosing Method: ManifestInfo()
File: src/main/scala/akka/util/ManifestInfo.scala:116
Taint Flags:

```
113
114 var manifests = Map.empty[String, Version]
115
116 try {
117 val resources = system.dynamicAccess.classLoader.getResources("META-INF/MANIFEST.MF")
118 while (resources.hasMoreElements()) {
119 val ios = resources.nextElement().openStream()
```

src/main/scala/akka/util/WildcardIndex.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$2
Enclosing Method: WildcardTree()
File: src/main/scala/akka/util/WildcardIndex.scala:41
Taint Flags:

```
38 }
39
40 private[akka] object WildcardTree {
41 private val empty = new WildcardTree[Nothing]()
42 def apply[T](): WildcardTree[T] = empty.asInstanceOf[WildcardTree[T]]
43 }
44
```

src/main/scala/akka/util/Version.scala, line 10 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: apply
Enclosing Method: Version()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

src/main/scala/akka/util/Version.scala, line 10 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: src/main/scala/akka/util/Version.scala:10
Taint Flags:

```
7 import akka.annotation.InternalApi
8
9 object Version {
10  val Zero: Version = Version("0.0.0")
11
12  private val Undefined = 0
13
```

src/main/scala/akka/util/MessageBuffer.scala, line 17 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: _head
Enclosing Method: MessageBuffer()
File: src/main/scala/akka/util/MessageBuffer.scala:17
Taint Flags:

```
14 final class MessageBuffer private (private var _head: MessageBuffer.Node, private var _tail: MessageBuffer.Node) {
15  import MessageBuffer._
16
17  private var _size: Int = if (_head eq null) 0 else 1
18
19  /**
20  * Check if the message buffer is empty.
```

src/main/scala/akka/util/WildcardIndex.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: $default$1
Enclosing Method: WildcardTree()
File: src/main/scala/akka/util/WildcardIndex.scala:41
Taint Flags:

```
38 }
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

src/main/scala/akka/util/WildcardIndex.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
39
40 private[akka] object WildcardTree {
41  private val empty = new WildcardTree[Nothing]()
42  def apply[T](): WildcardTree[T] = empty.asInstanceOf[WildcardTree[T]]
43 }
44
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 33 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: createLock
Enclosing Method: BoundedBlockingQueue()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:33
Taint Flags:

```
30  require(maxCapacity > 0)
31  }
32
33  protected val lock = createLock()
34  protected val notEmpty = createNotEmptyCondition()
35  protected val notFull = createNotFullCondition()
36
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 34 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: createNotEmptyCondition
Enclosing Method: BoundedBlockingQueue()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:34
Taint Flags:

```
31  }
32
33  protected val lock = createLock()
34  protected val notEmpty = createNotEmptyCondition()
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 34 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
35 protected val notFull = createNotFullCondition()
36
37 protected def createLock(): ReentrantLock = new ReentrantLock(false)
```

src/main/scala/akka/util/SegmentedRecencyList.scala, line 47 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: limits
Enclosing Method: SegmentedRecencyList()
File: src/main/scala/akka/util/SegmentedRecencyList.scala:47
Taint Flags:

```
44 import SegmentedRecencyList.Node
45
46 private var limits: immutable.IndexedSeq[Int] = initialLimits.toIndexedSeq
47 private var totalLimit: Int = limits.sum
48
49 private val levels = limits.size
50 private val lowest = 0
```

src/main/scala/akka/util/SegmentedRecencyList.scala, line 49 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: limits
Enclosing Method: SegmentedRecencyList()
File: src/main/scala/akka/util/SegmentedRecencyList.scala:49
Taint Flags:

```
46 private var limits: immutable.IndexedSeq[Int] = initialLimits.toIndexedSeq
47 private var totalLimit: Int = limits.sum
48
49 private val levels = limits.size
50 private val lowest = 0
51 private val highest = levels - 1
52
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

src/main/scala/akka/util/Helpers.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: toRootLowerCase
Enclosing Method: Helpers()
File: src/main/scala/akka/util/Helpers.scala:24
Taint Flags:

```
21
22 def toRootLowerCase(s: String): String = s.toLowerCase(Locale.ROOT)
23
24 val isWindows: Boolean = toRootLowerCase(System.getProperty("os.name", "")).indexOf("win") >= 0
25
26 def makePattern(s: String): Pattern =
27 Pattern.compile("^\\Q" + s.replace("?", "\\E.\\Q").replace("*", "\\E.*\\Q") + "\\E$")
```

src/main/scala/akka/util/ImmutableIntMap.scala, line 31 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: kvs
Enclosing Method: ImmutableIntMap()
File: src/main/scala/akka/util/ImmutableIntMap.scala:31
Taint Flags:

```
28
29 private final def this(key: Int, value: Int) = {
30 this(new Array[Int](2), 1)
31 kvs(0) = key
32 kvs(1) = value
33 }
34
```

src/main/scala/akka/util/ImmutableIntMap.scala, line 32 (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.util

src/main/scala/akka/util/ImmutableIntMap.scala, line 32 (Code Correctness: Constructor Invokes Overridable Function)	Low

Sink Details

Sink: FunctionCall: kvs
Enclosing Method: ImmutableIntMap()
File: src/main/scala/akka/util/ImmutableIntMap.scala:32
Taint Flags:

```
29 private final def this(key: Int, value: Int) = {
30 this(new Array[Int](2), 1)
31 kvs(0) = key
32 kvs(1) = value
33 }
34
35 private[this] final def indexForKey(key: Int): Int = {
```

Code Correctness: Double-Checked Locking (1 issue)

Abstract

Double-checked locking is an incorrect idiom that does not achieve the intended effect.

Explanation

Many talented individuals have spent a great deal of time pondering ways to make double-checked locking work in order to improve performance. None have succeeded. **Example 1:** At first blush it may seem that the following bit of code achieves thread safety while avoiding unnecessary synchronization.

```
if (fitz == null) {
  synchronized (this) {
    if (fitz == null) {
      fitz = new Fitzer();
    }
  }
}
return fitz;
```

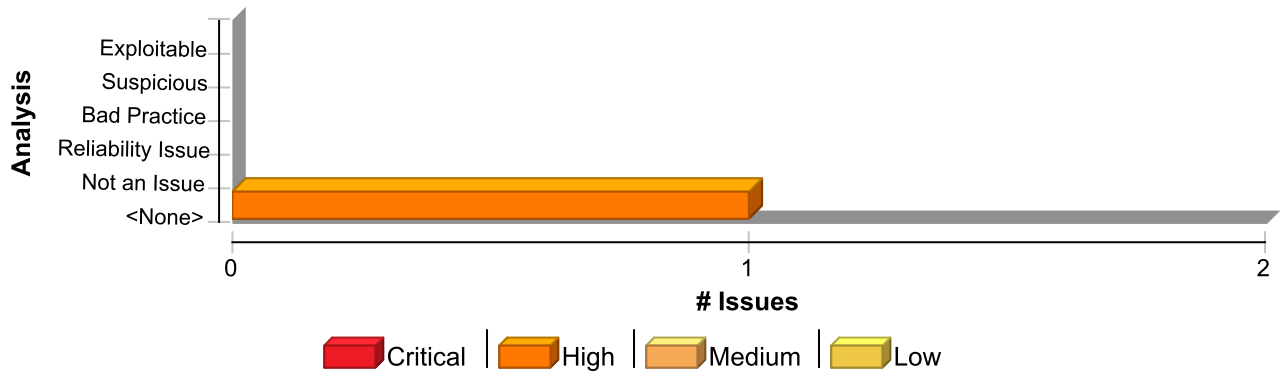
The programmer wants to guarantee that only one `Fitzer()` object is ever allocated, but does not want to pay the cost of synchronization every time this code is called. This idiom is known as double-checked locking. Unfortunately, it does not work, and multiple `Fitzer()` objects can be allocated. See The "Double-Checked Locking is Broken" Declaration for more details [1].

Recommendation

Synchronization is probably less expensive than you believe. In many cases, the best thing to do is to use the most straightforward solution. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
synchronized (this) {
  if (fitz == null) {
    fitz = new Fitzer();
  }
  return fitz;
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Double-Checked Locking	1	0	0	1
Total	**1**	**0**	**0**	**1**

Code Correctness: Double-Checked Locking — High

Package: akka.event

src/main/scala/akka/event/EventBus.scala, line 172 (Code Correctness: Double-Checked Locking) — High

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: publish()
File: src/main/scala/akka/event/EventBus.scala:172
Taint Flags:

```
169 if (cache contains c) cache(c) // c will never be removed from cache
170 else
171 subscriptions.synchronized {
172 if (cache contains c) cache(c)
173 else {
174 addToCache(subscriptions.addKey(c))
175 cache(c)
```

Code Correctness: Erroneous String Compare (58 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");
  }
```

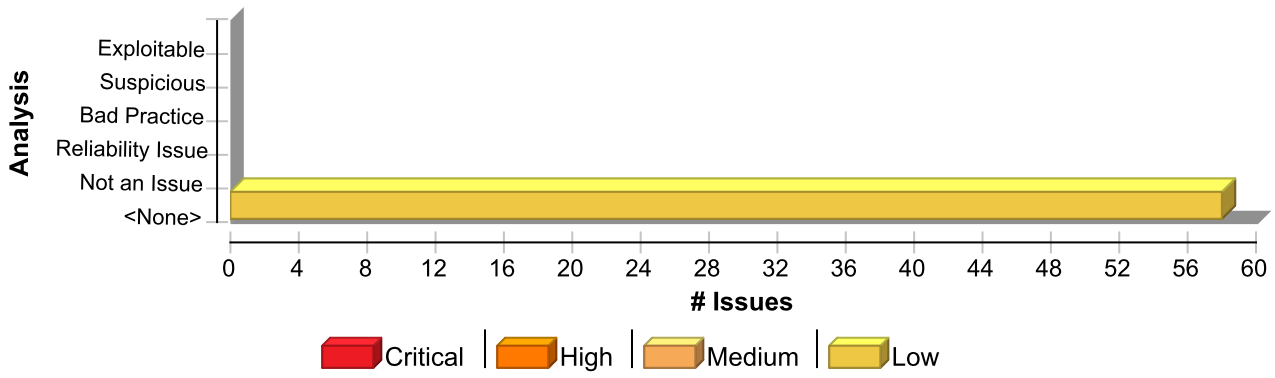
The `==` and `!=` operators will only behave as expected when they are used to compare strings contained in objects that are equal. The most common way for this to occur is for the strings to be interned, whereby the strings are added to a pool of objects maintained by the `String` class. Once a string 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	58	0	0	58
Total	**58**	**0**	**0**	**58**

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 453 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: ActorSystem$Settings()
File: src/main/scala/akka/actor/ActorSystem.scala:453
Taint Flags:

```
450 final val DebugUnhandledMessage: Boolean = getBoolean("akka.actor.debug.unhandled")
451 final val DebugRouterMisconfiguration: Boolean = getBoolean("akka.actor.debug.router-misconfiguration")
452
453 final val Home: Option[String] = config.getString("akka.home") match {
454 case "" => None
455 case x => Some(x)
456 }
```

src/main/scala/akka/actor/ActorSystem.scala, line 431 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: ActorSystem$Settings()
File: src/main/scala/akka/actor/ActorSystem.scala:431
Taint Flags:

```
428 final val LoggingFilter: String = getString("akka.logging-filter")
429 final val LoggerStartTimeout: Timeout = Timeout(config.getMillisDuration("akka.logger-startup-timeout"))
430 final val LogConfigOnStart: Boolean = config.getBoolean("akka.log-config-on-start")
431 final val LogDeadLetters: Int = toRootLowerCase(config.getString("akka.log-dead-letters")) match {
432 case "off" | "false" => 0
433 case "on" | "true" => Int.MaxValue
434 case _ => config.getInt("akka.log-dead-letters")
```

src/main/scala/akka/actor/TypedActor.scala, line 456 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

src/main/scala/akka/actor/TypedActor.scala, line 456 (Code Correctness: Erroneous String Compare) — Low

Sink Details

Sink: Operation
Enclosing Method: invoke()
File: src/main/scala/akka/actor/TypedActor.scala:456
Taint Flags:

```
453
454 def actor = actorVar.get
455 @throws(classOf[Throwable])
456 def invoke(proxy: AnyRef, method: Method, args: Array[AnyRef]): AnyRef = method.getName match {
457 case "toString" => actor.toString
458 case "equals" =>
459 (args.length == 1 && (proxy eq args(0)) || actor == extension.getActorRefFor(args(0)))
```

src/main/scala/akka/actor/ActorSystem.scala, line 431 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: ActorSystem$Settings()
File: src/main/scala/akka/actor/ActorSystem.scala:431
Taint Flags:

```
428 final val LoggingFilter: String = getString("akka.logging-filter")
429 final val LoggerStartTimeout: Timeout = Timeout(config.getMillisDuration("akka.logger-startup-timeout"))
430 final val LogConfigOnStart: Boolean = config.getBoolean("akka.log-config-on-start")
431 final val LogDeadLetters: Int = toRootLowerCase(config.getString("akka.log-dead-letters")) match {
432 case "off" | "false" => 0
433 case "on" | "true" => Int.MaxValue
434 case _ => config.getInt("akka.log-dead-letters")
```

src/main/scala/akka/actor/TypedActor.scala, line 456 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: invoke()

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

src/main/scala/akka/actor/TypedActor.scala, line 456 (Code Correctness: Erroneous String Compare) — Low

File: src/main/scala/akka/actor/TypedActor.scala:456
Taint Flags:

```
453
454 def actor = actorVar.get
455 @throws(classOf[Throwable])
456 def invoke(proxy: AnyRef, method: Method, args: Array[AnyRef]): AnyRef = method.getName match {
457 case "toString" => actor.toString
458 case "equals" =>
459 (args.length == 1 && (proxy eq args(0)) || actor == extension.getActorRefFor(args(0)))
```

src/main/scala/akka/actor/RepointableActorRef.scala, line 157 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getChild()
File: src/main/scala/akka/actor/RepointableActorRef.scala:157
Taint Flags:

```
154
155 def getChild(name: Iterator[String]): InternalActorRef =
156 if (name.hasNext) {
157 name.next() match {
158 case ".." => getParent.getChild(name)
159 case "" => getChild(name)
160 case other =>
```

src/main/scala/akka/actor/Props.scala, line 181 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: withMailbox()
File: src/main/scala/akka/actor/Props.scala:181
Taint Flags:

```
178 /**
```

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

src/main/scala/akka/actor/Props.scala, line 181 (Code Correctness: Erroneous String Compare) — Low

```
179 * Returns a new Props with the specified mailbox set.
180 */
181 def withMailbox(m: String): Props = deploy.mailbox match {
182 case NoMailboxGiven => copy(deploy = deploy.copy(mailbox = m))
183 case x => if (x == m) this else copy(deploy = deploy.copy(mailbox = m))
184 }
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 537 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getSingleChild()
File: src/main/scala/akka/actor/ActorRefProvider.scala:537
Taint Flags:

```
534 theOneWhoWalksTheBubblesOfSpaceTime,
535 rootPath) {
536 override def getParent: InternalActorRef = this
537 override def getSingleChild(name: String): InternalActorRef = name match {
538 case "temp" => tempContainer
539 case "deadLetters" => deadLetters
540 case other => extraNames.getOrElse(other, super.getSingleChild(other))
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 537 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getSingleChild()
File: src/main/scala/akka/actor/ActorRefProvider.scala:537
Taint Flags:

```
534 theOneWhoWalksTheBubblesOfSpaceTime,
535 rootPath) {
536 override def getParent: InternalActorRef = this
537 override def getSingleChild(name: String): InternalActorRef = name match {
```

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRefProvider.scala, line 537 (Code Correctness: Erroneous String Compare) — Low

```
538 case "temp" => tempContainer
539 case "deadLetters" => deadLetters
540 case other => extraNames.getOrElse(other, super.getSingleChild(other))
```

src/main/scala/akka/actor/ActorSystem.scala, line 431 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: ActorSystem$Settings()
File: src/main/scala/akka/actor/ActorSystem.scala:431
Taint Flags:

```
428 final val LoggingFilter: String = getString("akka.logging-filter")
429 final val LoggerStartTimeout: Timeout = Timeout(config.getMillisDuration("akka.logger-startup-timeout"))
430 final val LogConfigOnStart: Boolean = config.getBoolean("akka.log-config-on-start")
431 final val LogDeadLetters: Int = toRootLowerCase(config.getString("akka.log-dead-letters")) match {
432 case "off" | "false" => 0
433 case "on" | "true" => Int.MaxValue
434 case _ => config.getInt("akka.log-dead-letters")
```

src/main/scala/akka/actor/Props.scala, line 150 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: dispatcher()
File: src/main/scala/akka/actor/Props.scala:150
Taint Flags:

```
147 * Convenience method for extracting the dispatcher information from the
148 * contained [[Deploy]] instance.
149 */
150 def dispatcher: String = deploy.dispatcher match {
151 case NoDispatcherGiven => Dispatchers.DefaultDispatcherId
152 case x => x
153 }
```

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

src/main/scala/akka/actor/Props.scala, line 173 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: withDispatcher()
File: src/main/scala/akka/actor/Props.scala:173
Taint Flags:

```
170 /**
171 * Returns a new Props with the specified dispatcher set.
172 */
173 def withDispatcher(d: String): Props = deploy.dispatcher match {
174 case NoDispatcherGiven => copy(deploy = deploy.copy(dispatcher = d))
175 case x => if (x == d) this else copy(deploy = deploy.copy(dispatcher = d))
176 }
```

src/main/scala/akka/actor/TypedActor.scala, line 456 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: invoke()
File: src/main/scala/akka/actor/TypedActor.scala:456
Taint Flags:

```
453
454 def actor = actorVar.get
455 @throws(classOf[Throwable])
456 def invoke(proxy: AnyRef, method: Method, args: Array[AnyRef]): AnyRef = method.getName match {
457 case "toString" => actor.toString
458 case "equals" =>
459 (args.length == 1 && (proxy eq args(0)) || actor == extension.getActorRefFor(args(0)))
```

src/main/scala/akka/actor/ActorSystem.scala, line 431 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 431 (Code Correctness: Erroneous String Compare) — Low

Sink Details

Sink: Operation
Enclosing Method: ActorSystem$Settings()
File: src/main/scala/akka/actor/ActorSystem.scala:431
Taint Flags:

```
428 final val LoggingFilter: String = getString("akka.logging-filter")
429 final val LoggerStartTimeout: Timeout = Timeout(config.getMillisDuration("akka.logger-startup-timeout"))
430 final val LogConfigOnStart: Boolean = config.getBoolean("akka.log-config-on-start")
431 final val LogDeadLetters: Int = toRootLowerCase(config.getString("akka.log-dead-letters")) match {
432 case "off" | "false" => 0
433 case "on" | "true" => Int.MaxValue
434 case _ => config.getInt("akka.log-dead-letters")
```

src/main/scala/akka/actor/Props.scala, line 159 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: mailbox()
File: src/main/scala/akka/actor/Props.scala:159
Taint Flags:

```
156 * Convenience method for extracting the mailbox information from the
157 * contained [[Deploy]] instance.
158 */
159 def mailbox: String = deploy.mailbox match {
160 case NoMailboxGiven => Mailboxes.DefaultMailboxId
161 case x => x
162 }
```

src/main/scala/akka/actor/ActorSystem.scala, line 439 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: ActorSystem$Settings()

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 439 (Code Correctness: Erroneous String Compare) — Low

File: src/main/scala/akka/actor/ActorSystem.scala:439
Taint Flags:

```
436 final val LogDeadLettersDuringShutdown: Boolean = config.getBoolean("akka.log-dead-letters-during-shutdown")
437 final val LogDeadLettersSuspendDuration: Duration = {
438 val key = "akka.log-dead-letters-suspend-duration"
439 toRootLowerCase(config.getString(key)) match {
440 case "infinite" => Duration.Inf
441 case _ => config.getMillisDuration(key)
442 }
```

src/main/scala/akka/actor/RepointableActorRef.scala, line 157 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getChild()
File: src/main/scala/akka/actor/RepointableActorRef.scala:157
Taint Flags:

```
154
155 def getChild(name: Iterator[String]): InternalActorRef =
156 if (name.hasNext) {
157 name.next() match {
158 case ".." => getParent.getChild(name)
159 case "" => getChild(name)
160 case other =>
```

src/main/scala/akka/actor/ActorRef.scala, line 427 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: rec()
File: src/main/scala/akka/actor/ActorRef.scala:427
Taint Flags:

```
424 def rec(ref: InternalActorRef, name: Iterator[String]): InternalActorRef =
```

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 427 (Code Correctness: Erroneous String Compare) — Low

```
425 ref match {
426 case l: LocalActorRef =>
427 val next = name.next() match {
428 case ".." => l.getParent
429 case "" => l
430 case any => l.getSingleChild(any)
```

src/main/scala/akka/actor/ActorRef.scala, line 427 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: rec()
File: src/main/scala/akka/actor/ActorRef.scala:427
Taint Flags:

```
424 def rec(ref: InternalActorRef, name: Iterator[String]): InternalActorRef =
425 ref match {
426 case l: LocalActorRef =>
427 val next = name.next() match {
428 case ".." => l.getParent
429 case "" => l
430 case any => l.getSingleChild(any)
```

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/Children.scala, line 252 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: checkName()
File: src/main/scala/akka/actor/dungeon/Children.scala:252
Taint Flags:

```
249 */
250
251 private def checkName(name: String): String = {
```

Code Correctness: Erroneous String Compare — Low

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/Children.scala, line 252 (Code Correctness: Erroneous String Compare) — Low

```
252 name match {
253 case null => throw InvalidActorNameException("actor name must not be null")
254 case "" => throw InvalidActorNameException("actor name must not be empty")
255 case _ =>
```

Package: akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 359 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: configurator()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:359
Taint Flags:

```
356 def dispatcher(): MessageDispatcher
357
358 def configureExecutor(): ExecutorServiceConfigurator = {
359 def configurator(executor: String): ExecutorServiceConfigurator = executor match {
360 case null | "" | "fork-join-executor" =>
361 new ForkJoinExecutorConfigurator(config.getConfig("fork-join-executor"), prerequisites)
362 case "thread-pool-executor" =>
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 359 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: configurator()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:359
Taint Flags:

```
356 def dispatcher(): MessageDispatcher
357
358 def configureExecutor(): ExecutorServiceConfigurator = {
359 def configurator(executor: String): ExecutorServiceConfigurator = executor match {
360 case null | "" | "fork-join-executor" =>
```

Code Correctness: Erroneous String Compare — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 359 (Code Correctness: Erroneous String Compare) — Low

```
361 new ForkJoinExecutorConfigurator(config.getConfig("fork-join-executor"), prerequisites)
362 case "thread-pool-executor" =>
```

src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala, line 94 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: createExecutorServiceFactory()
File: src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala:94
Taint Flags:

```
91 case other => other
92 }
93
94 val asyncMode = config.getString("task-peeking-mode") match {
95 case "FIFO" => true
96 case "LIFO" => false
97 case _ =>
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 230 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: lookupConfigurator()
File: src/main/scala/akka/dispatch/Mailboxes.scala:230
Taint Flags:

```
227 if (!settings.config.hasPath(id)) throw new ConfigurationException(s"Mailbox Type [$id] not configured")
228 val conf = config(id)
229
230 val mailboxType = conf.getString("mailbox-type") match {
231 case "" => throw new ConfigurationException(s"The setting mailbox-type, defined in [$id] is empty")
232 case fqcn =>
233 val args = List(classOf[ActorSystem.Settings] -> settings, classOf[Config] -> conf)
```

Code Correctness: Erroneous String Compare — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Dispatchers.scala, line 256 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: configuratorFrom()
File: src/main/scala/akka/dispatch/Dispatchers.scala:256
Taint Flags:

```
253 if (!cfg.hasPath("id"))
254 throw new ConfigurationException("Missing dispatcher 'id' property in config: " + cfg.root.render)
255
256 cfg.getString("type") match {
257 case "Dispatcher" => new DispatcherConfigurator(cfg, prerequisites)
258 case "BalancingDispatcher" =>
259 // FIXME remove this case in 2.4
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 216 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: lookupConfigurator()
File: src/main/scala/akka/dispatch/Mailboxes.scala:216
Taint Flags:

```
213 mailboxTypeConfigurators.get(id) match {
214 case null =>
215 // It doesn't matter if we create a mailbox type configurator that isn't used due to concurrent lookup.
216 val newConfigurator = id match {
217 // TODO RK remove these two for Akka 2.3
218 case "unbounded" => UnboundedMailbox()
219 case "bounded" => new BoundedMailbox(settings, config(id))
```

src/main/scala/akka/dispatch/Dispatchers.scala, line 256 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Erroneous String Compare — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Dispatchers.scala, line 256 (Code Correctness: Erroneous String Compare) — Low

Sink Details

Sink: Operation
Enclosing Method: configuratorFrom()
File: src/main/scala/akka/dispatch/Dispatchers.scala:256
Taint Flags:

```
253 if (!cfg.hasPath("id"))
254 throw new ConfigurationException("Missing dispatcher 'id' property in config: " + cfg.root.render)
255
256 cfg.getString("type") match {
257 case "Dispatcher" => new DispatcherConfigurator(cfg, prerequisites)
258 case "BalancingDispatcher" =>
259 // FIXME remove this case in 2.4
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 359 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: configurator()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:359
Taint Flags:

```
356 def dispatcher(): MessageDispatcher
357
358 def configureExecutor(): ExecutorServiceConfigurator = {
359 def configurator(executor: String): ExecutorServiceConfigurator = executor match {
360 case null | "" | "fork-join-executor" =>
361 new ForkJoinExecutorConfigurator(config.getConfig("fork-join-executor"), prerequisites)
362 case "thread-pool-executor" =>
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 382 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: configureExecutor()

Code Correctness: Erroneous String Compare — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 382 (Code Correctness: Erroneous String Compare) — Low

File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:382
Taint Flags:

```
379 .get
380 }
381
382 config.getString("executor") match {
383 case "default-executor" =>
384 new DefaultExecutorServiceConfigurator(
385 config.getConfig("default-executor"),
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 123 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getMailboxRequirement()
File: src/main/scala/akka/dispatch/Mailboxes.scala:123
Taint Flags:

```
120 private var mailboxSizeWarningIssued = false
121 private var mailboxNonZeroPushTimeoutWarningIssued = false
122
123 def getMailboxRequirement(config: Config) = config.getString("mailbox-requirement") match {
124 case NoMailboxRequirement => classOf[MessageQueue]
125 case x => dynamicAccess.getClassFor[AnyRef](x).get
126 }
```

src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala, line 94 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: createExecutorServiceFactory()
File: src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala:94
Taint Flags:

```
91 case other => other
```

Code Correctness: Erroneous String Compare — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala, line 94 (Code Correctness: Erroneous String Compare) — Low

```
92  }
93
94  val asyncMode = config.getString("task-peeking-mode") match {
95  case "FIFO" => true
96  case "LIFO" => false
97  case _ =>
```

src/main/scala/akka/dispatch/Dispatchers.scala, line 256 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: configuratorFrom()
File: src/main/scala/akka/dispatch/Dispatchers.scala:256
Taint Flags:

```
253  if (!cfg.hasPath("id"))
254  throw new ConfigurationException("Missing dispatcher 'id' property in config: " + cfg.root.render)
255
256  cfg.getString("type") match {
257  case "Dispatcher" => new DispatcherConfigurator(cfg, prerequisites)
258  case "BalancingDispatcher" =>
259  // FIXME remove this case in 2.4
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 359 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: configurator()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:359
Taint Flags:

```
356  def dispatcher(): MessageDispatcher
357
358  def configureExecutor(): ExecutorServiceConfigurator = {
359  def configurator(executor: String): ExecutorServiceConfigurator = executor match {
```

Code Correctness: Erroneous String Compare — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 359 (Code Correctness: Erroneous String Compare) — Low

```
360 case null | "" | "fork-join-executor" =>
361 new ForkJoinExecutorConfigurator(config.getConfig("fork-join-executor"), prerequisites)
362 case "thread-pool-executor" =>
```

src/main/scala/akka/dispatch/Mailboxes.scala, line 216 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: lookupConfigurator()
File: src/main/scala/akka/dispatch/Mailboxes.scala:216
Taint Flags:

```
213 mailboxTypeConfigurators.get(id) match {
214 case null =>
215 // It doesn't matter if we create a mailbox type configurator that isn't used due to concurrent lookup.
216 val newConfigurator = id match {
217 // TODO RK remove these two for Akka 2.3
218 case "unbounded" => UnboundedMailbox()
219 case "bounded" => new BoundedMailbox(settings, config(id))
```

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 507 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: levelFor()
File: src/main/scala/akka/event/Logging.scala:507
Taint Flags:

```
504 * valid inputs are upper or lowercase (not mixed) versions of:
505 * "error", "warning", "info" and "debug"
506 */
507 def levelFor(s: String): Option[LogLevel] = Helpers.toRootLowerCase(s) match {
508 case "off" => Some(OffLevel)
509 case "error" => Some(ErrorLevel)
```

Code Correctness: Erroneous String Compare — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 507 (Code Correctness: Erroneous String Compare) — Low

```
510 case "warning" => Some(WarningLevel)
```

src/main/scala/akka/event/Logging.scala, line 507 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: levelFor()
File: src/main/scala/akka/event/Logging.scala:507
Taint Flags:

```
504 * valid inputs are upper or lowercase (not mixed) versions of:
505 * "error", "warning", "info" and "debug"
506 */
507 def levelFor(s: String): Option[LogLevel] = Helpers.toRootLowerCase(s) match {
508 case "off" => Some(OffLevel)
509 case "error" => Some(ErrorLevel)
510 case "warning" => Some(WarningLevel)
```

src/main/scala/akka/event/Logging.scala, line 507 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: levelFor()
File: src/main/scala/akka/event/Logging.scala:507
Taint Flags:

```
504 * valid inputs are upper or lowercase (not mixed) versions of:
505 * "error", "warning", "info" and "debug"
506 */
507 def levelFor(s: String): Option[LogLevel] = Helpers.toRootLowerCase(s) match {
508 case "off" => Some(OffLevel)
509 case "error" => Some(ErrorLevel)
510 case "warning" => Some(WarningLevel)
```

Code Correctness: Erroneous String Compare — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 507 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: levelFor()
File: src/main/scala/akka/event/Logging.scala:507
Taint Flags:

```
504 * valid inputs are upper or lowercase (not mixed) versions of:
505 * "error", "warning", "info" and "debug"
506 */
507 def levelFor(s: String): Option[LogLevel] = Helpers.toRootLowerCase(s) match {
508 case "off" => Some(OffLevel)
509 case "error" => Some(ErrorLevel)
510 case "warning" => Some(WarningLevel)
```

src/main/scala/akka/event/Logging.scala, line 507 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: levelFor()
File: src/main/scala/akka/event/Logging.scala:507
Taint Flags:

```
504 * valid inputs are upper or lowercase (not mixed) versions of:
505 * "error", "warning", "info" and "debug"
506 */
507 def levelFor(s: String): Option[LogLevel] = Helpers.toRootLowerCase(s) match {
508 case "off" => Some(OffLevel)
509 case "error" => Some(ErrorLevel)
510 case "warning" => Some(WarningLevel)
```

Package: akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 88 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality

Code Correctness: Erroneous String Compare — Low

Package: akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 88 (Code Correctness: Erroneous String Compare) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getTtl()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:88
Taint Flags:

```
85 }
86
87 private def getTtl(path: String, positive: Boolean): CachePolicy =
88 config.getString(path) match {
89 case "default" => if (positive) defaultCachePolicy else defaultNegativeCachePolicy
90 case "forever" => Forever
91 case "never" => Never
```

src/main/scala/akka/io/Tcp.scala, line 619 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: TcpExt$Settings()
File: src/main/scala/akka/io/Tcp.scala:619
Taint Flags:

```
616 }
617 val ManagementDispatcher: String = getString("management-dispatcher")
618 val FileIODispatcher: String = getString("file-io-dispatcher")
619 val TransferToLimit: Int = getString("file-io-transferTo-limit") match {
620 case "unlimited" => Int.MaxValue
621 case _ => getIntBytes("file-io-transferTo-limit")
622 }
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 88 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation

Code Correctness: Erroneous String Compare — Low

Package: akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 88 (Code Correctness: Erroneous String Compare) — Low

Enclosing Method: getTtl()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:88
Taint Flags:

```
85 }
86
87 private def getTtl(path: String, positive: Boolean): CachePolicy =
88 config.getString(path) match {
89 case "default" => if (positive) defaultCachePolicy else defaultNegativeCachePolicy
90 case "forever" => Forever
91 case "never" => Never
```

src/main/scala/akka/io/Tcp.scala, line 609 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: TcpExt$Settings()
File: src/main/scala/akka/io/Tcp.scala:609
Taint Flags:

```
606 val BatchAcceptLimit: Int = getInt("batch-accept-limit").requiring(_ > 0, "batch-accept-limit must be > 0")
607 val DirectBufferSize: Int = getIntBytes("direct-buffer-size")
608 val MaxDirectBufferPoolSize: Int = getInt("direct-buffer-pool-limit")
609 val RegisterTimeout: Duration = getString("register-timeout") match {
610 case "infinite" => Duration.Undefined
611 case _ => _config.getMillisDuration("register-timeout")
612 }
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 88 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getTtl()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:88
Taint Flags:

```
85 }
```

Code Correctness: Erroneous String Compare — Low

Package: akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 88 (Code Correctness: Erroneous String Compare) — Low

```
86
87  private def getTtl(path: String, positive: Boolean): CachePolicy =
88  config.getString(path) match {
89  case "default" => if (positive) defaultCachePolicy else defaultNegativeCachePolicy
90  case "forever" => Forever
91  case "never" => Never
```

src/main/scala/akka/io/SelectionHandler.scala, line 32 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: SelectionHandlerSettings()
File: src/main/scala/akka/io/SelectionHandler.scala:32
Taint Flags:

```
29 abstract class SelectionHandlerSettings(config: Config) {
30  import config._
31
32  val MaxChannels: Int = getString("max-channels") match {
33  case "unlimited" => -1
34  case _ => getInt("max-channels").requiring(_ > 0, "max-channels must be > 0 or 'unlimited'")
35  }
```

src/main/scala/akka/io/Dns.scala, line 253 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getInetAddress()
File: src/main/scala/akka/io/Dns.scala:253
Taint Flags:

```
250 */
251 @InternalApi
252 def getInetAddress(ipv4: Option[Inet4Address], ipv6: Option[Inet6Address]): Option[InetAddress] =
253 System.getProperty("java.net.preferIPv6Addresses") match {
254 case "true" => ipv6.orElse(ipv4)
```

Code Correctness: Erroneous String Compare — Low

Package: akka.io

src/main/scala/akka/io/Dns.scala, line 253 (Code Correctness: Erroneous String Compare) — Low

```
255 case _ => ipv4.orElse(ipv6)
256 }
```

src/main/scala/akka/io/Tcp.scala, line 613 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: TcpExt$Settings()
File: src/main/scala/akka/io/Tcp.scala:613
Taint Flags:

```
610 case "infinite" => Duration.Undefined
611 case _ => _config.getMillisDuration("register-timeout")
612 }
613 val ReceivedMessageSizeLimit: Int = getString("max-received-message-size") match {
614 case "unlimited" => Int.MaxValue
615 case _ => getIntBytes("max-received-message-size")
616 }
```

src/main/scala/akka/io/Tcp.scala, line 629 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: TcpExt$Settings()
File: src/main/scala/akka/io/Tcp.scala:629
Taint Flags:

```
626 getInt("finish-connect-retries").requiring(_ > 0, "finish-connect-retries must be > 0")
627
628 val WindowsConnectionAbortWorkaroundEnabled
629 : Boolean = getString("windows-connection-abort-workaround-enabled") match {
630 case "auto" => Helpers.isWindows
631 case _ => getBoolean("windows-connection-abort-workaround-enabled")
632 }
```

src/main/scala/akka/io/TcpConnection.scala, line 360 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Code Correctness: Erroneous String Compare — Low

Package: akka.io

src/main/scala/akka/io/TcpConnection.scala, line 360 (Code Correctness: Erroneous String Compare) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: extractMsg()
File: src/main/scala/akka/io/TcpConnection.scala:360
Taint Flags:

```
357 @tailrec private[this] def extractMsg(t: Throwable): String =
358 if (t == null) "unknown"
359 else {
360 t.getMessage match {
361 case null | "" => extractMsg(t.getCause)
362 case msg => msg
363 }
```

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 61 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getTtl()
File: src/main/scala/akka/io/dns/DnsSettings.scala:61
Taint Flags:

```
58 val NegativeCachePolicy: CachePolicy = getTtl("negative-ttl")
59
60 private def getTtl(path: String): CachePolicy =
61 c.getString(path) match {
62 case "forever" => Forever
63 case "never" => Never
64 case _ =>
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 38 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Erroneous String Compare — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 38 (Code Correctness: Erroneous String Compare) — Low

Sink Details

Sink: Operation
Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:38
Taint Flags:

```
35 val NameServers: List[InetSocketAddress] = {
36 c.getValue("nameservers").valueType() match {
37 case ConfigValueType.STRING =>
38 c.getString("nameservers") match {
39 case "default" =>
40 val osAddresses = getDefaultNameServers(system).getOrElse(failUnableToDetermineDefaultNameservers)
41 if (osAddresses.isEmpty) failUnableToDetermineDefaultNameservers
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 93 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:93
Taint Flags:

```
90 val SearchDomains: List[String] = {
91 c.getValue("search-domains").valueType() match {
92 case ConfigValueType.STRING =>
93 c.getString("search-domains") match {
94 case "default" => resolvConf.map(_.search).getOrElse(Nil)
95 case single => List(single)
96 }
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 61 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getTtl()

Code Correctness: Erroneous String Compare — Low

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 61 (Code Correctness: Erroneous String Compare) — Low

File: src/main/scala/akka/io/dns/DnsSettings.scala:61
Taint Flags:

```
58 val NegativeCachePolicy: CachePolicy = getTtl("negative-ttl")
59
60 private def getTtl(path: String): CachePolicy =
61 c.getString(path) match {
62 case "forever" => Forever
63 case "never" => Never
64 case _ =>
```

src/main/scala/akka/io/dns/DnsSettings.scala, line 106 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: DnsSettings()
File: src/main/scala/akka/io/dns/DnsSettings.scala:106
Taint Flags:

```
103 val NDots: Int = {
104 c.getValue("ndots").valueType() match {
105 case ConfigValueType.STRING =>
106 c.getString("ndots") match {
107 case "default" => resolvConf.map(_.ndots).getOrElse(1)
108 case _ =>
109 throw new IllegalArgumentException("Invalid value for ndots. Must be the string 'default' or an integer.")
```

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/Deployer.scala, line 249 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: apply()
File: src/main/scala/akka/actor/Deployer.scala:249
Taint Flags:

Code Correctness: Erroneous String Compare — Low

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/Deployer.scala, line 249 (Code Correctness: Erroneous String Compare) — Low

```
246 def deploy(d: Deploy): Unit = {
247 @tailrec def add(path: Array[String], d: Deploy): Unit = {
248 val w: WildcardIndex[Deploy] = deployments.get
249 for (i <- path.indices) path(i) match {
250 case "" => throw InvalidActorNameException(s"Actor name in deployment [${d.path}] must not be empty")
251 case el => ActorPath.validatePathElement(el, fullPath = d.path)
252 }
```

Package: src.main.scala.akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 409 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: apply()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:409
Taint Flags:

```
406 .configure(Some(config.getInt("task-queue-size")).flatMap {
407 case size if size > 0 =>
408 Some(config.getString("task-queue-type"))
409 .map {
410 case "array" => ThreadPoolConfig.arrayBlockingQueue(size, false) //TODO config fairness?
411 case "" | "linked" => ThreadPoolConfig.linkedBlockingQueue(size)
412 case x =>
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 409 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: apply()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:409
Taint Flags:

```
406 .configure(Some(config.getInt("task-queue-size")).flatMap {
407 case size if size > 0 =>
```

Code Correctness: Erroneous String Compare	Low

Package: src.main.scala.akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 409 (Code Correctness: Erroneous String Compare)	Low

```
408 Some(config.getString("task-queue-type"))
409 .map {
410 case "array" => ThreadPoolConfig.arrayBlockingQueue(size, false) //TODO config fairness?
411 case "" | "linked" => ThreadPoolConfig.linkedBlockingQueue(size)
412 case x =>
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 409 (Code Correctness: Erroneous String Compare)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: apply()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:409
Taint Flags:

```
406 .configure(Some(config.getInt("task-queue-size")).flatMap {
407 case size if size > 0 =>
408 Some(config.getString("task-queue-type"))
409 .map {
410 case "array" => ThreadPoolConfig.arrayBlockingQueue(size, false) //TODO config fairness?
411 case "" | "linked" => ThreadPoolConfig.linkedBlockingQueue(size)
412 case x =>
```

Code Correctness: Incorrect Serializable Method Signature (2 issues)

Abstract

Using the incorrect method signature on a method used in serialization may lead to it never being called.

Explanation

Code Correctness: Incorrect Serializable Method Signature issues occur when a serializable class creates a serialization or deserialization function but does not follow the correct signatures:

```
  private void writeObject(java.io.ObjectOutputStream out) throws IOException;
  private void readObject(java.io.ObjectInputStream in) throws IOException,
ClassNotFoundException;
  private void readObjectNoData() throws ObjectStreamException;
```

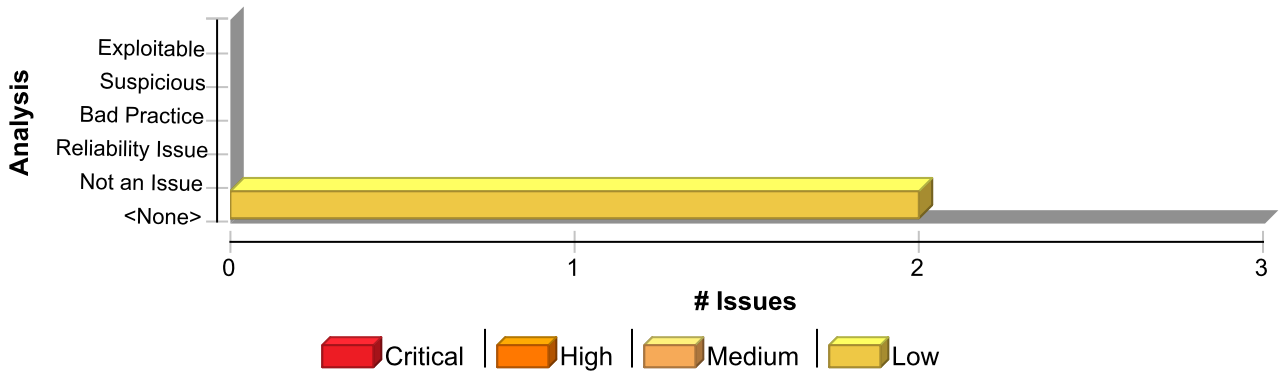
Deviating from the method signatures that serialization requires may mean that the method is never called during serialization/deserialization, leading to incomplete serialization/deserialization, or could mean that untrusted code could gain access to the objects. In the case that there are exceptions that are not thrown, it may mean that serialization/deserialization fails and crashes the application or potentially even fails quietly such that objects may be only partially constructed correctly, leading to flaws that can be extremely difficult to debug. The caller should catch these exceptions such that incorrect serialization/deserialization can be handled properly without a crash or partially constructed objects.

Recommendation

When using serialization in Java for classes that require special handling, the `writeObject()`, `readObject()` and `readObjectNoData` methods must have the exact signatures:

```
  private void writeObject(java.io.ObjectOutputStream out) throws IOException;
  private void readObject(java.io.ObjectInputStream in) throws IOException,
ClassNotFoundException;
  private void readObjectNoData() throws ObjectStreamException;
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Incorrect Serializable Method Signature	2	0	0	2
Total	**2**	**0**	**0**	**2**

Code Correctness: Incorrect Serializable Method Signature — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 713 (Code Correctness: Incorrect Serializable Method Signature) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: writeObject
Enclosing Method: writeObject()
File: src/main/scala-2.13/akka/util/ByteString.scala:713
Taint Flags:

```
710
711 @SerialVersionUID(1L)
712 private class SerializationProxy(@transient private var orig: ByteString) extends Serializable {
713 private def writeObject(out: ObjectOutputStream): Unit = {
714 out.writeByte(orig.byteStringCompanion.SerializationIdentity)
715 orig.writeToOutputStream(out)
716 }
```

src/main/scala-2.13/akka/util/ByteString.scala, line 718 (Code Correctness: Incorrect Serializable Method Signature) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: readObject
Enclosing Method: readObject()
File: src/main/scala-2.13/akka/util/ByteString.scala:718
Taint Flags:

```
715 orig.writeToOutputStream(out)
716 }
717
718 private def readObject(in: ObjectInputStream): Unit = {
719 val serializationId = in.readByte()
720
721 orig = Companion(from = serializationId).readFromInputStream(in)
```

Code Correctness: Non-Static Inner Class Implements Serializable (77 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

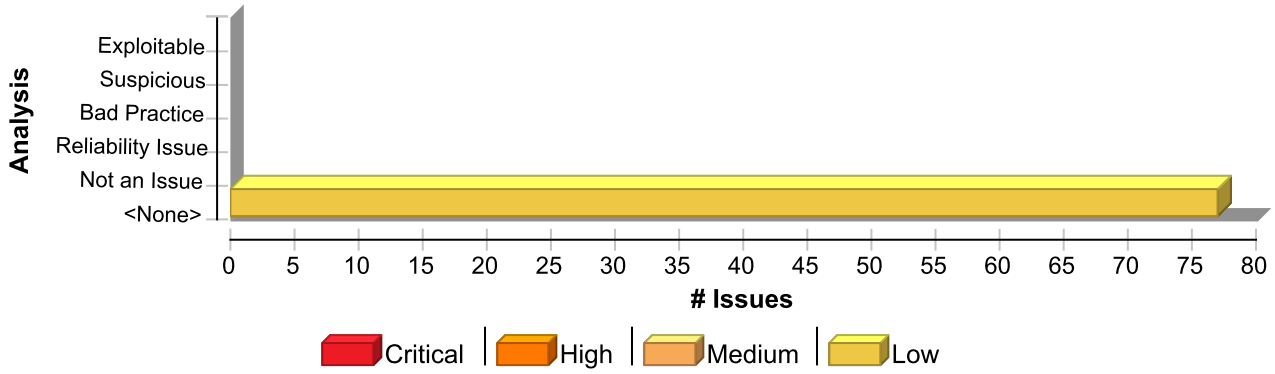
When using inner classes, they should not be serialized, or they should be changed to static-nested classes, since these do not have the drawbacks that non-static inner classes have when serialized. When a nested class is static it inherently has no association with instance variables (including those 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	77	0	0	77
Total	**77**	**0**	**0**	**77**

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

Package: akka.actor

src/main/scala/akka/actor/FSM.scala, line 331 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: FSM$Event
File: src/main/scala/akka/actor/FSM.scala:331
Taint Flags:

```
328 * All messages sent to the [[akka.actor.FSM]] will be wrapped inside an
329 * `Event`, which allows pattern matching to extract both state and data.
330 */
331 final case class Event[D](event: Any, stateData: D) extends NoSerializationVerificationNeeded
332
333 /**
334 * Case class representing the state of the [[akka.actor.FSM]] within the
```

src/main/scala/akka/actor/FSM.scala, line 168 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: FSM$SilentState

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

Package: akka.actor

src/main/scala/akka/actor/FSM.scala, line 168 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

File: src/main/scala/akka/actor/FSM.scala:168
Taint Flags:

```
165 * INTERNAL API
166 * Using a subclass for binary compatibility reasons
167 */
168 private[akka] class SilentState[S, D](
169 stateName: S,
170 stateData: D,
171 timeout: Option[FiniteDuration],
```

src/main/scala/akka/actor/FSM.scala, line 211 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: FSM$State
File: src/main/scala/akka/actor/FSM.scala:211
Taint Flags:

```
208 Some((state.stateName, state.stateData, state.timeout, state.stopReason, state.replies))
209 }
210 }
211 class State[S, D](
212 val stateName: S,
213 val stateData: D,
214 val timeout: Option[FiniteDuration] = None,
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 910 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: CoordinatedShutdownTerminationWatcher$WatchedTimedOut
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:910
Taint Flags:

```
907 @InternalApi
908 private[akka] object CoordinatedShutdownTerminationWatcher {
909 final case class Watch(actor: ActorRef, deadline: Deadline, completionPromise: Promise[Done])
```

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

Package: akka.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 910 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
910 private final case class WatchedTimedOut(actor: ActorRef, completionPromise: Promise[Done], timeout: FiniteDuration)
911
912 def props: Props = Props(new CoordinatedShutdownTerminationWatcher)
913 }
```

src/main/scala/akka/actor/ActorSystem.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: ProviderSelection$Custom
File: src/main/scala/akka/actor/ActorSystem.scala:107
Taint Flags:

```
104 // these two cannot be referenced by class as they may not be on the classpath
105 case object Remote extends ProviderSelection("remote", RemoteActorRefProvider, hasCluster = false)
106 case object Cluster extends ProviderSelection("cluster", ClusterActorRefProvider, hasCluster = true)
107 final case class Custom(override val fqcn: String) extends ProviderSelection("custom", fqcn, hasCluster = false)
108
109 /**
110 * JAVA API
```

src/main/scala/akka/actor/FSM.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: FSM$Timer
File: src/main/scala/akka/actor/FSM.scala:122
Taint Flags:

```
119 * INTERNAL API
120 */
121 @InternalApi
122 private[akka] final case class Timer(name: String, msg: Any, mode: TimerMode, generation: Int, owner: AnyRef)(
123 context: ActorContext)
124 extends NoSerializationVerificationNeeded {
125 private var ref: Option[Cancellable] = _
```

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

Package: akka.actor

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

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: FSM$Transition
File: src/main/scala/akka/actor/FSM.scala:46
Taint Flags:

```
43 * Message type which is used to communicate transitions between states to
44 * all subscribed listeners (use [[akka.actor.FSM.SubscribeTransitionCallBack]]).
45 */
46 final case class Transition[S](fsmRef: ActorRef, from: S, to: S)
47
48 /**
49 * Send this to an [[akka.actor.FSM]] to request first the [[FSM.CurrentState]]
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 292 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: CoordinatedShutdown$Phase
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:292
Taint Flags:

```
289 /**
290 * INTERNAL API
291 */
292 private[akka] final case class Phase(
293 dependsOn: Set[String],
294 timeout: FiniteDuration,
295 recover: Boolean,
```

src/main/scala/akka/actor/FSM.scala, line 92 (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.actor

src/main/scala/akka/actor/FSM.scala, line 92 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: FSM$TimeoutMarker
File: src/main/scala/akka/actor/FSM.scala:92
Taint Flags:

```
89 /**
90 * INTERNAL API
91 */
92 private final case class TimeoutMarker(generation: Long)
93
94 /** INTERNAL API */
95 @InternalApi
```

src/main/scala/akka/actor/ActorRef.scala, line 646 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: DeadLetterActorRef$SerializedDeadLetterActorRef
File: src/main/scala/akka/actor/ActorRef.scala:646
Taint Flags:

```
643
644 private[akka] object DeadLetterActorRef {
645 @SerialVersionUID(1L)
646 class SerializedDeadLetterActorRef extends Serializable { //TODO implement as Protobuf for performance?
647 @throws(classOf[java.io.ObjectStreamException])
648 private def readResolve(): AnyRef = JavaSerializer.currentSystem.value.deadLetters
649 }
```

src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.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: TimerSchedulerImpl$Timer
File: src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.scala:23
Taint Flags:

```
20 def owner: TimerSchedulerImpl
21 }
```

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

Package: akka.actor

src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.scala, line 23 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
22
23  final case class Timer(key: Any, msg: Any, repeat: Boolean, generation: Int, task: Cancellable)
24  final case class InfluenceReceiveTimeoutTimerMsg(key: Any, generation: Int, owner: TimerSchedulerImpl)
25  extends TimerMsg
26  with NoSerializationVerificationNeeded
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 341 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LightArrayRevolverScheduler$TaskQueue
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:341
Taint Flags:

```
338 object LightArrayRevolverScheduler {
339  private[this] val taskOffset = unsafe.objectFieldOffset(classOf[TaskHolder].getDeclaredField("task"))
340
341  private class TaskQueue extends AbstractNodeQueue[TaskHolder]
342
343  /**
344  * INTERNAL API
```

src/main/scala/akka/actor/FSM.scala, line 159 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: FSM$LogEntry
File: src/main/scala/akka/actor/FSM.scala:159
Taint Flags:

```
156  /**
157  * Log Entry of the [[akka.actor.LoggingFSM]], can be obtained by calling `getLog`.
158  */
159  final case class LogEntry[S, D](stateName: S, stateData: D, event: Any)
160
161  /** Used by `forMax` to signal "cancel stateTimeout" */
162  private final val SomeMaxFiniteDuration = Some(Long.MaxValue.nanos)
```

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

Package: akka.actor

src/main/scala/akka/actor/FSM.scala, line 159 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.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: TimerSchedulerImpl$InfluenceReceiveTimeoutTimerMsg
File: src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.scala:24
Taint Flags:

```
21 }
22
23 final case class Timer(key: Any, msg: Any, repeat: Boolean, generation: Int, task: Cancellable)
24 final case class InfluenceReceiveTimeoutTimerMsg(key: Any, generation: Int, owner: TimerSchedulerImpl)
25 extends TimerMsg
26 with NoSerializationVerificationNeeded
27 final case class NotInfluenceReceiveTimeoutTimerMsg(key: Any, generation: Int, owner: TimerSchedulerImpl)
```

src/main/scala/akka/actor/FSM.scala, line 82 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: FSM$Failure
File: src/main/scala/akka/actor/FSM.scala:82
Taint Flags:

```
79 * an error, e.g. if the state to transition into does not exist. You can use
80 * this to communicate a more precise cause to the `onTermination` block.
81 */
82 final case class Failure(cause: Any) extends Reason
83
84 /**
85 * This case object is received in case of a state timeout.
```

src/main/scala/akka/actor/FSM.scala, line 337 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

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

Package: akka.actor

src/main/scala/akka/actor/FSM.scala, line 337 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: FSM$StopEvent
File: src/main/scala/akka/actor/FSM.scala:337
Taint Flags:

```
334 * Case class representing the state of the [[akka.actor.FSM]] within the
335 * `onTermination` block.
336 */
337 final case class StopEvent[S, D](reason: Reason, currentState: S, stateData: D)
338 extends NoSerializationVerificationNeeded
339
340 }
```

src/main/scala/akka/actor/FSM.scala, line 53 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: FSM$SubscribeTransitionCallBack
File: src/main/scala/akka/actor/FSM.scala:53
Taint Flags:

```
50 * and then a series of [[FSM.Transition]] updates. Cancel the subscription
51 * using [[FSM.UnsubscribeTransitionCallBack]].
52 */
53 final case class SubscribeTransitionCallBack(actorRef: ActorRef)
54
55 /**
56 * Unsubscribe from [[akka.actor.FSM.Transition]] notifications which was
```

src/main/scala/akka/actor/FSM.scala, line 40 (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.actor

src/main/scala/akka/actor/FSM.scala, line 40 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: FSM$CurrentState
File: src/main/scala/akka/actor/FSM.scala:40
Taint Flags:

```
37 * [[akka.actor.FSM.SubscribeTransitionCallBack]] before sending any
38 * [[akka.actor.FSM.Transition]] messages.
39 */
40 final case class CurrentState[S](fsmRef: ActorRef, state: S)
41
42 /**
43 * Message type which is used to communicate transitions between states to
```

src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.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: TimerSchedulerImpl$FixedDelayMode
File: src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.scala:38
Taint Flags:

```
35 private case class FixedRateMode(initialDelay: FiniteDuration) extends TimerMode {
36 override def repeat: Boolean = true
37 }
38 private case class FixedDelayMode(initialDelay: FiniteDuration) extends TimerMode {
39 override def repeat: Boolean = true
40 }
41 private case object SingleMode extends TimerMode {
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 909 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: CoordinatedShutdownTerminationWatcher$Watch
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:909
Taint Flags:

```
906 /** INTERNAL API */
907 @InternalApi
```

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

Package: akka.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 909 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
908 private[akka] object CoordinatedShutdownTerminationWatcher {
909   final case class Watch(actor: ActorRef, deadline: Deadline, completionPromise: Promise[Done])
910   private final case class WatchedTimedOut(actor: ActorRef, completionPromise: Promise[Done], timeout: FiniteDuration)
911
912   def props: Props = Props(new CoordinatedShutdownTerminationWatcher)
```

src/main/scala/akka/actor/FSM.scala, line 59 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: FSM$UnsubscribeTransitionCallBack
File: src/main/scala/akka/actor/FSM.scala:59
Taint Flags:

```
56 * Unsubscribe from [[akka.actor.FSM.Transition]] notifications which was
57 * effected by sending the corresponding [[akka.actor.FSM.SubscribeTransitionCallBack]].
58 */
59 final case class UnsubscribeTransitionCallBack(actorRef: ActorRef)
60
61 /**
62 * Reason why this [[akka.actor.FSM]] is shutting down.
```

src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.scala, line 35 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TimerSchedulerImpl$FixedRateMode
File: src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.scala:35
Taint Flags:

```
32 private sealed trait TimerMode {
33 def repeat: Boolean
34 }
35 private case class FixedRateMode(initialDelay: FiniteDuration) extends TimerMode {
36 override def repeat: Boolean = true
37 }
38 private case class FixedDelayMode(initialDelay: FiniteDuration) extends TimerMode {
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.actor

src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.scala, line 35 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.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: TimerSchedulerImpl$NotInfluenceReceiveTimeoutTimerMsg
File: src/main/scala/akka/actor/dungeon/TimerSchedulerImpl.scala:27
Taint Flags:

```
24 final case class InfluenceReceiveTimeoutTimerMsg(key: Any, generation: Int, owner: TimerSchedulerImpl)
25 extends TimerMsg
26 with NoSerializationVerificationNeeded
27 final case class NotInfluenceReceiveTimeoutTimerMsg(key: Any, generation: Int, owner: TimerSchedulerImpl)
28 extends TimerMsg
29 with NoSerializationVerificationNeeded
30 with NotInfluenceReceiveTimeout
```

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/ChildrenContainer.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: ChildrenContainer$Creation
File: src/main/scala/akka/actor/dungeon/ChildrenContainer.scala:49
Taint Flags:

```
46 case object UserRequest extends SuspendReason
47 // careful with those system messages, all handling to be taking place in ActorCell.scala!
48 final case class Recreation(cause: Throwable) extends SuspendReason with WaitingForChildren
49 final case class Creation() extends SuspendReason with WaitingForChildren
50 case object Termination extends SuspendReason
51
52 class ChildRestartsIterable(stats: immutable.Map[_, ChildStats])
```

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

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/ChildrenContainer.scala, line 48 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ChildrenContainer$Recreation
File: src/main/scala/akka/actor/dungeon/ChildrenContainer.scala:48
Taint Flags:

```
45 sealed trait SuspendReason
46 case object UserRequest extends SuspendReason
47 // careful with those system messages, all handling to be taking place in ActorCell.scala!
48 final case class Recreation(cause: Throwable) extends SuspendReason with WaitingForChildren
49 final case class Creation() extends SuspendReason with WaitingForChildren
50 case object Termination extends SuspendReason
51
```

src/main/scala/akka/actor/dungeon/ChildrenContainer.scala, line 163 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ChildrenContainer$TerminatingChildrenContainer
File: src/main/scala/akka/actor/dungeon/ChildrenContainer.scala:163
Taint Flags:

```
160 * type of container, depending on whether or not children are left and whether or not
161 * the reason was “Terminating”.
162 */
163 final case class TerminatingChildrenContainer(
164 c: immutable.TreeMap[String, ChildStats],
165 toDie: Set[ActorRef],
166 reason: SuspendReason)
```

src/main/scala/akka/actor/dungeon/FaultHandling.scala, line 30 (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.actor.dungeon

src/main/scala/akka/actor/dungeon/FaultHandling.scala, line 30 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: FaultHandling$FailedRef
File: src/main/scala/akka/actor/dungeon/FaultHandling.scala:30
Taint Flags:

```
27 @InternalApi private[akka] object FaultHandling {
28  sealed trait FailedInfo
29  private case object NoFailedInfo extends FailedInfo
30  private final case class FailedRef(ref: ActorRef) extends FailedInfo
31  private case object FailedFatally extends FailedInfo
32 }
33
```

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 835 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: UnboundedDequeBasedMailbox$MessageQueue
File: src/main/scala/akka/dispatch/Mailbox.scala:835
Taint Flags:

```
832 }
833
834 object UnboundedDequeBasedMailbox {
835  class MessageQueue extends LinkedBlockingDeque[Envelope] with UnboundedDequeBasedMessageQueue {
836  final val queue = this
837  }
838 }
```

src/main/scala/akka/dispatch/Mailbox.scala, line 659 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: UnboundedMailbox$MessageQueue
File: src/main/scala/akka/dispatch/Mailbox.scala:659
Taint Flags:

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

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 659 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
656 }
657
658 object UnboundedMailbox {
659  class MessageQueue extends ConcurrentLinkedQueue[Envelope] with UnboundedQueueBasedMessageQueue {
660  final def queue: Queue[Envelope] = this
661  }
662 }
```

src/main/scala/akka/dispatch/BalancingDispatcher.scala, line 66 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BalancingDispatcher$SharingMailbox
File: src/main/scala/akka/dispatch/BalancingDispatcher.scala:66
Taint Flags:

```
63 */
64 private[akka] val messageQueue: MessageQueue = _mailboxType.create(None, None)
65
66 private class SharingMailbox(val system: ActorSystemImpl, _messageQueue: MessageQueue)
67 extends Mailbox(_messageQueue)
68 with DefaultSystemMessageQueue {
69 override def cleanUp(): Unit = {
```

src/main/scala/akka/dispatch/BatchingExecutor.scala, line 92 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BatchingExecutor$BlockableBatch
File: src/main/scala/akka/dispatch/BatchingExecutor.scala:92
Taint Flags:

```
89
90 private[this] val _blockContext = new ThreadLocal[BlockContext]()
91
92 private[this] final class BlockableBatch extends AbstractBatch with BlockContext {
93 // this method runs in the delegate ExecutionContext's thread
```

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

Package: akka.dispatch

src/main/scala/akka/dispatch/BatchingExecutor.scala, line 92 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
94 override final def run(): Unit = {
95 require(_tasksLocal.get eq null)
```

src/main/scala/akka/dispatch/Mailbox.scala, line 738 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: UnboundedPriorityMailbox$MessageQueue
File: src/main/scala/akka/dispatch/Mailbox.scala:738
Taint Flags:

```
735 }
736
737 object UnboundedPriorityMailbox {
738 class MessageQueue(initialCapacity: Int, cmp: Comparator[Envelope])
739 extends PriorityBlockingQueue[Envelope](initialCapacity, cmp)
740 with UnboundedQueueBasedMessageQueue {
741 final def queue: Queue[Envelope] = this
```

src/main/scala/akka/dispatch/Mailbox.scala, line 920 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: UnboundedControlAwareMailbox$MessageQueue
File: src/main/scala/akka/dispatch/Mailbox.scala:920
Taint Flags:

```
917 }
918
919 object UnboundedControlAwareMailbox {
920 class MessageQueue extends UnboundedControlAwareMessageQueueSemantics with java.io.Serializable {
921 val controlQueue: Queue[Envelope] = new ConcurrentLinkedQueue[Envelope]()
922 val queue: Queue[Envelope] = new ConcurrentLinkedQueue[Envelope]()
923 }
```

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

Package: akka.dispatch

src/main/scala/akka/dispatch/BatchingExecutor.scala, line 60 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BatchingExecutor$AbstractBatch
File: src/main/scala/akka/dispatch/BatchingExecutor.scala:60
Taint Flags:

```
57 // invariant: if "_tasksLocal.get ne null" then we are inside Batch.run; if it is null, we are outside
58 private[this] val _tasksLocal = new ThreadLocal[AbstractBatch]()
59
60 private[this] abstract class AbstractBatch extends ArrayDeque[Runnable](4) with Runnable {
61 @tailrec final def processBatch(batch: AbstractBatch): Unit =
62 if ((batch eq this) && !batch.isEmpty) {
63 batch.poll().run()
```

src/main/scala/akka/dispatch/Mailbox.scala, line 718 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BoundedMailbox$MessageQueue
File: src/main/scala/akka/dispatch/Mailbox.scala:718
Taint Flags:

```
715 }
716
717 object BoundedMailbox {
718 class MessageQueue(capacity: Int, final val pushTimeOut: FiniteDuration)
719 extends LinkedBlockingQueue[Envelope](capacity)
720 with BoundedQueueBasedMessageQueue {
721 final def queue: BlockingQueue[Envelope] = this
```

src/main/scala/akka/dispatch/CachingConfig.scala, line 26 (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.dispatch

src/main/scala/akka/dispatch/CachingConfig.scala, line 26 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: CachingConfig$ValuePathEntry
File: src/main/scala/akka/dispatch/CachingConfig.scala:26
Taint Flags:

```
23 val exists: Boolean
24 val config: Config
25 }
26 final case class ValuePathEntry(valid: Boolean, exists: Boolean, config: Config = emptyConfig) extends PathEntry
27 final case class StringPathEntry(valid: Boolean, exists: Boolean, config: Config, value: String) extends PathEntry
28
29 val invalidPathEntry = ValuePathEntry(false, true)
```

src/main/scala/akka/dispatch/CachingConfig.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: CachingConfig$StringPathEntry
File: src/main/scala/akka/dispatch/CachingConfig.scala:27
Taint Flags:

```
24 val config: Config
25 }
26 final case class ValuePathEntry(valid: Boolean, exists: Boolean, config: Config = emptyConfig) extends PathEntry
27 final case class StringPathEntry(valid: Boolean, exists: Boolean, config: Config, value: String) extends PathEntry
28
29 val invalidPathEntry = ValuePathEntry(false, true)
30 val nonExistingPathEntry = ValuePathEntry(true, false)
```

src/main/scala/akka/dispatch/BatchingExecutor.scala, line 77 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BatchingExecutor$Batch
File: src/main/scala/akka/dispatch/BatchingExecutor.scala:77
Taint Flags:

```
74 }
75 }
```

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

Package: akka.dispatch

src/main/scala/akka/dispatch/BatchingExecutor.scala, line 77 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
76
77  private[this] final class Batch extends AbstractBatch {
78  override final def run: Unit = {
79  require(_tasksLocal.get eq null)
80  _tasksLocal.set(this) // Install ourselves as the current batch
```

src/main/scala/akka/dispatch/Mailbox.scala, line 942 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BoundedControlAwareMailbox$MessageQueue
File: src/main/scala/akka/dispatch/Mailbox.scala:942
Taint Flags:

```
939 }
940
941 object BoundedControlAwareMailbox {
942  class MessageQueue(val capacity: Int, val pushTimeOut: FiniteDuration)
943  extends BoundedControlAwareMessageQueueSemantics
944  with java.io.Serializable {
945
```

src/main/scala/akka/dispatch/Mailbox.scala, line 860 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BoundedDequeBasedMailbox$MessageQueue
File: src/main/scala/akka/dispatch/Mailbox.scala:860
Taint Flags:

```
857 }
858
859 object BoundedDequeBasedMailbox {
860  class MessageQueue(capacity: Int, val pushTimeOut: FiniteDuration)
861  extends LinkedBlockingDeque[Envelope](capacity)
862  with BoundedDequeBasedMessageQueue {
863  final val queue = this
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 860 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala, line 43 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ForkJoinExecutorConfigurator$AkkaForkJoinTask
File: src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala:43
Taint Flags:

```
40 * INTERNAL AKKA USAGE ONLY
41 */
42 @SerialVersionUID(1L)
43 final class AkkaForkJoinTask(runnable: Runnable) extends ForkJoinTask[Unit] {
44 override def getRawResult(): Unit = ()
45 override def setRawResult(unit: Unit): Unit = ()
46 override def exec(): Boolean =
```

Package: akka.event

src/main/scala/akka/event/ActorClassificationUnsubscriber.scala, line 76 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ActorClassificationUnsubscriber$Unregister
File: src/main/scala/akka/event/ActorClassificationUnsubscriber.scala:76
Taint Flags:

```
73 private val unsubscribersCount = new AtomicInteger(0)
74
75 final case class Register(actor: ActorRef, seq: Int)
76 final case class Unregister(actor: ActorRef, seq: Int)
77
78 def start(
79 system: ActorSystem,
```

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

Package: akka.event

src/main/scala/akka/event/ActorClassificationUnsubscriber.scala, line 75 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ActorClassificationUnsubscriber$Register
File: src/main/scala/akka/event/ActorClassificationUnsubscriber.scala:75
Taint Flags:

```
72
73  private val unsubscribersCount = new AtomicInteger(0)
74
75  final case class Register(actor: ActorRef, seq: Int)
76  final case class Unregister(actor: ActorRef, seq: Int)
77
78  def start(
```

src/main/scala/akka/event/EventStreamUnsubscriber.scala, line 69 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: EventStreamUnsubscriber$Register
File: src/main/scala/akka/event/EventStreamUnsubscriber.scala:69
Taint Flags:

```
66
67  private val unsubscribersCount = new AtomicInteger(0)
68
69  final case class Register(actor: ActorRef)
70
71  final case class UnregisterIfNoMoreSubscribedChannels(actor: ActorRef)
72
```

src/main/scala/akka/event/EventStreamUnsubscriber.scala, line 71 (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.event

src/main/scala/akka/event/EventStreamUnsubscriber.scala, line 71 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: EventStreamUnsubscriber$UnregisterIfNoMoreSubscribedChannels
File: src/main/scala/akka/event/EventStreamUnsubscriber.scala:71
Taint Flags:

```
68
69 final case class Register(actor: ActorRef)
70
71 final case class UnregisterIfNoMoreSubscribedChannels(actor: ActorRef)
72
73 private def props(eventStream: EventStream, debug: Boolean) =
74 Props(classOf[EventStreamUnsubscriber], eventStream, debug).withDispatcher(Dispatchers.InternalDispatcherId)
```

Package: akka.io

src/main/scala/akka/io/Dns.scala, line 75 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Dns$Resolved
File: src/main/scala/akka/io/Dns.scala:75
Taint Flags:

```
72 }
73
74 @deprecated("Use cached(DnsProtocol.Resolved)", "2.6.0")
75 case class Resolved(name: String, ipv4: immutable.Seq[Inet4Address], ipv6: immutable.Seq[Inet6Address])
76 extends Command {
77 val addrOption: Option[InetAddress] = IpVersionSelector.getInetAddress(ipv4.headOption, ipv6.headOption)
78
```

src/main/scala/akka/io/SimpleDnsCache.scala, line 134 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: SimpleDnsCache$CacheEntry
File: src/main/scala/akka/io/SimpleDnsCache.scala:134
Taint Flags:

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

Package: akka.io

src/main/scala/akka/io/SimpleDnsCache.scala, line 134 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
131  }
132  }
133
134  private[io] case class CacheEntry[T](answer: T, until: Long) {
135  def isValid(clock: Long): Boolean = clock < until
136  }
137
```

src/main/scala/akka/io/TcpListener.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: TcpListener$RegisterIncoming
File: src/main/scala/akka/io/TcpListener.scala:23
Taint Flags:

```
20  */
21 private[io] object TcpListener {
22
23  final case class RegisterIncoming(channel: SocketChannel)
24  extends HasFailureMessage
25  with NoSerializationVerificationNeeded {
26  def failureMessage = FailedRegisterIncoming(channel)
```

src/main/scala/akka/io/SelectionHandler.scala, line 86 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: SelectionHandler$WorkerForCommand
File: src/main/scala/akka/io/SelectionHandler.scala:86
Taint Flags:

```
83  def failureMessage: Any
84  }
85
86  final case class WorkerForCommand(
87  apiCommand: HasFailureMessage,
```

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

Package: akka.io

src/main/scala/akka/io/SelectionHandler.scala, line 86 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
88 commander: ActorRef,
89 childProps: ChannelRegistry => Props)
```

src/main/scala/akka/io/TcpConnection.scala, line 554 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TcpConnection$CloseInformation
File: src/main/scala/akka/io/TcpConnection.scala:554
Taint Flags:

```
551 * Used to transport information to the postStop method to notify
552 * interested party about a connection close.
553 */
554 final case class CloseInformation(notificationsTo: Set[ActorRef], closedEvent: Event)
555
556 /**
557 * Groups required connection-related data that are only available once the connection has been fully established.
```

src/main/scala/akka/io/TcpListener.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: TcpListener$FailedRegisterIncoming
File: src/main/scala/akka/io/TcpListener.scala:29
Taint Flags:

```
26 def failureMessage = FailedRegisterIncoming(channel)
27 }
28
29 final case class FailedRegisterIncoming(channel: SocketChannel) extends NoSerializationVerificationNeeded
30
31 }
32
```

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

Package: akka.io

src/main/scala/akka/io/TcpConnection.scala, line 559 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TcpConnection$ConnectionInfo
File: src/main/scala/akka/io/TcpConnection.scala:559
Taint Flags:

```
556 /**
557 * Groups required connection-related data that are only available once the connection has been fully established.
558 */
559 final case class ConnectionInfo(
560 registration: ChannelRegistration,
561 handler: ActorRef,
562 keepOpenOnPeerClosed: Boolean,
```

src/main/scala/akka/io/SelectionHandler.scala, line 92 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: SelectionHandler$Retry
File: src/main/scala/akka/io/SelectionHandler.scala:92
Taint Flags:

```
89 childProps: ChannelRegistry => Props)
90 extends NoSerializationVerificationNeeded
91
92 final case class Retry(command: WorkerForCommand, retriesLeft: Int) extends NoSerializationVerificationNeeded {
93 require(retriesLeft >= 0)
94 }
95
```

src/main/scala/akka/io/Dns.scala, line 70 (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.io

src/main/scala/akka/io/Dns.scala, line 70 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: Dns$Resolve
File: src/main/scala/akka/io/Dns.scala:70
Taint Flags:

```
67 sealed trait Command
68
69 @deprecated("Use cached(DnsProtocol.Resolve)", "2.6.0")
70 case class Resolve(name: String) extends Command with ConsistentHashable {
71 override def consistentHashKey = name
72 }
73
```

src/main/scala/akka/io/TcpConnection.scala, line 567 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TcpConnection$UpdatePendingWriteAndThen
File: src/main/scala/akka/io/TcpConnection.scala:567
Taint Flags:

```
564
565 // INTERNAL MESSAGES
566
567 final case class UpdatePendingWriteAndThen(remainingWrite: PendingWrite, work: () => Unit)
568 extends NoSerializationVerificationNeeded
569 final case class WriteFileFailed(e: IOException)
570 case object Unregistered
```

src/main/scala/akka/io/TcpConnection.scala, line 569 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TcpConnection$WriteFileFailed
File: src/main/scala/akka/io/TcpConnection.scala:569
Taint Flags:

```
566
567 final case class UpdatePendingWriteAndThen(remainingWrite: PendingWrite, work: () => Unit)
```

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

Package: akka.io

src/main/scala/akka/io/TcpConnection.scala, line 569 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
568 extends NoSerializationVerificationNeeded
569 final case class WriteFileFailed(e: IOException)
570 case object Unregistered
571
572 sealed abstract class PendingWrite {
```

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 249 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AsyncDnsResolver$ResolveFailedException
File: src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala:249
Taint Flags:

```
246 private val Empty =
247 Future.successful(Answer(-1, immutable.Seq.empty[ResourceRecord], immutable.Seq.empty[ResourceRecord]))
248
249 case class ResolveFailedException(msg: String) extends Exception(msg)
250 }
251
252 undefined
```

src/main/scala/akka/io/dns/internal/DnsClient.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: DnsClient$Answer
File: src/main/scala/akka/io/dns/internal/DnsClient.scala:32
Taint Flags:

```
29 final case class SrvQuestion(id: Short, name: String) extends DnsQuestion
30 final case class Question4(id: Short, name: String) extends DnsQuestion
31 final case class Question6(id: Short, name: String) extends DnsQuestion
32 final case class Answer(id: Short, rrs: im.Seq[ResourceRecord], additionalRecs: im.Seq[ResourceRecord] = Nil)
33 extends NoSerializationVerificationNeeded
34 final case class DropRequest(id: Short)
```

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

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsClient.scala, line 32 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
35 }
```

src/main/scala/akka/io/dns/internal/DnsClient.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: DnsClient$Question6
File: src/main/scala/akka/io/dns/internal/DnsClient.scala:31
Taint Flags:

```
28 }
29 final case class SrvQuestion(id: Short, name: String) extends DnsQuestion
30 final case class Question4(id: Short, name: String) extends DnsQuestion
31 final case class Question6(id: Short, name: String) extends DnsQuestion
32 final case class Answer(id: Short, rrs: im.Seq[ResourceRecord], additionalRecs: im.Seq[ResourceRecord] = Nil)
33 extends NoSerializationVerificationNeeded
34 final case class DropRequest(id: Short)
```

src/main/scala/akka/io/dns/internal/DnsClient.scala, line 34 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: DnsClient$DropRequest
File: src/main/scala/akka/io/dns/internal/DnsClient.scala:34
Taint Flags:

```
31 final case class Question6(id: Short, name: String) extends DnsQuestion
32 final case class Answer(id: Short, rrs: im.Seq[ResourceRecord], additionalRecs: im.Seq[ResourceRecord] = Nil)
33 extends NoSerializationVerificationNeeded
34 final case class DropRequest(id: Short)
35 }
36
37 /**
```

src/main/scala/akka/io/dns/internal/DnsClient.scala, line 30 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

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

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsClient.scala, line 30 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: DnsClient$Question4
File: src/main/scala/akka/io/dns/internal/DnsClient.scala:30
Taint Flags:

```
27 def id: Short
28 }
29 final case class SrvQuestion(id: Short, name: String) extends DnsQuestion
30 final case class Question4(id: Short, name: String) extends DnsQuestion
31 final case class Question6(id: Short, name: String) extends DnsQuestion
32 final case class Answer(id: Short, rrs: im.Seq[ResourceRecord], additionalRecs: im.Seq[ResourceRecord] = Nil)
33 extends NoSerializationVerificationNeeded
```

src/main/scala/akka/io/dns/internal/DnsClient.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: DnsClient$SrvQuestion
File: src/main/scala/akka/io/dns/internal/DnsClient.scala:29
Taint Flags:

```
26 sealed trait DnsQuestion {
27 def id: Short
28 }
29 final case class SrvQuestion(id: Short, name: String) extends DnsQuestion
30 final case class Question4(id: Short, name: String) extends DnsQuestion
31 final case class Question6(id: Short, name: String) extends DnsQuestion
32 final case class Answer(id: Short, rrs: im.Seq[ResourceRecord], additionalRecs: im.Seq[ResourceRecord] = Nil)
```

Package: akka.japi

src/main/scala/akka/japi/JavaAPI.scala, line 99 (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.japi

src/main/scala/akka/japi/JavaAPI.scala, line 99 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: JavaPartialFunction$NoMatchException
File: src/main/scala/akka/japi/JavaAPI.scala:99
Taint Flags:

```
96 }
97
98 object JavaPartialFunction {
99  sealed abstract class NoMatchException extends RuntimeException with NoStackTrace
100  case object NoMatch extends NoMatchException
101  final def noMatch(): RuntimeException = NoMatch
102 }
```

src/main/scala/akka/japi/JavaAPI.scala, line 207 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Option$Some
File: src/main/scala/akka/japi/JavaAPI.scala:207
Taint Flags:

```
204  * Class <code>Some[A]</code> represents existing values of type
205  * <code>A</code>.
206  */
207  final case class Some[A](v: A) extends Option[A] {
208  def get: A = v
209  def getOrElse[B >: A](defaultValue: B): B = v
210  def isEmpty: Boolean = false
```

Package: akka.pattern

src/main/scala/akka/pattern/BackoffSupervisor.scala, line 293 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BackoffSupervisor$RestartCount
File: src/main/scala/akka/pattern/BackoffSupervisor.scala:293
Taint Flags:

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

Package: akka.pattern

src/main/scala/akka/pattern/BackoffSupervisor.scala, line 293 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
290 */
291 def getRestartCount = GetRestartCount
292
293 final case class RestartCount(count: Int)
294
295 /**
296 * INTERNAL API
```

src/main/scala/akka/pattern/StatusReply.scala, line 102 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: StatusReply$ErrorMessage
File: src/main/scala/akka/pattern/StatusReply.scala:102
Taint Flags:

```
99 *
100 * Not meant for usage outside of [[StatusReply]].
101 */
102 final case class ErrorMessage(private val errorMessage: String)
103 extends RuntimeException(errorMessage)
104 with NoStackTrace {
105 override def toString: String = errorMessage
```

src/main/scala/akka/pattern/BackoffSupervisor.scala, line 305 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BackoffSupervisor$ResetRestartCount
File: src/main/scala/akka/pattern/BackoffSupervisor.scala:305
Taint Flags:

```
302 * INTERNAL API
303 */
304 @InternalApi
305 private[akka] case class ResetRestartCount(current: Int) extends DeadLetterSuppression
306
```

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

Package: akka.pattern

src/main/scala/akka/pattern/BackoffSupervisor.scala, line 305 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
307 /**
308 * INTERNAL API
```

src/main/scala/akka/pattern/AskSupport.scala, line 703 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: PromiseActorRef$StoppedWithPath
File: src/main/scala/akka/pattern/AskSupport.scala:703
Taint Flags:

```
700 private[akka] object PromiseActorRef {
701 private case object Registering
702 private case object Stopped
703 private final case class StoppedWithPath(path: ActorPath)
704
705 private val ActorStopResult = Failure(ActorKilledException("Stopped"))
706 private val defaultOnTimeout: String => Throwable = str => new AskTimeoutException(str)
```

src/main/scala/akka/pattern/BackoffSupervisor.scala, line 261 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BackoffSupervisor$CurrentChild
File: src/main/scala/akka/pattern/BackoffSupervisor.scala:261
Taint Flags:

```
258 * Send this message to the `BackoffSupervisor` and it will reply with
259 * [[BackoffSupervisor.CurrentChild]] containing the `ActorRef` of the current child, if any.
260 */
261 final case class CurrentChild(ref: Option[ActorRef]) {
262
263 /**
264 * Java API: The `ActorRef` of the current child, if any
```

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

Package: akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 91 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Serialization$Information
File: src/main/scala/akka/serialization/Serialization.scala:91
Taint Flags:

```
88  * or if serializer library e.g. custom serializer/deserializer in Jackson need
89  * access to the current `ActorSystem`.
90  */
91  final case class Information(address: Address, system: ActorSystem)
92
93  /**
94  * Sets serialization information in a `ThreadLocal` and runs `f`. The information is
```

src/main/scala/akka/serialization/Serializer.scala, line 404 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: DisabledJavaSerializer$JavaSerializationException
File: src/main/scala/akka/serialization/Serializer.scala:404
Taint Flags:

```
401 }
402
403 object DisabledJavaSerializer {
404  final class JavaSerializationException(msg: String) extends RuntimeException(msg) with NoStackTrace
405  final val IllegalSerialization = new JavaSerializationException(
406  "Attempted to serialize message using Java serialization while `akka.actor.allow-java-serialization` was disabled. Check WARNING logs for more details.")
407  final val IllegalDeserialization = new JavaSerializationException(
```

Package: akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 510 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

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

Package: akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 510 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink Details

Sink: Class: ByteString$ByteStrings
File: src/main/scala-2.13/akka/util/ByteString.scala:510
Taint Flags:

```
507 /**
508 * A ByteString with 2 or more fragments.
509 */
510 final class ByteStrings private (private[akka] val bytestrings: Vector[ByteString1], val length: Int)
511 extends ByteString
512 with Serializable {
513 if (bytestrings.isEmpty) throw new IllegalArgumentException("bytestrings must not be empty")
```

src/main/scala-2.13/akka/util/ByteString.scala, line 158 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ByteString$ByteString1C
File: src/main/scala-2.13/akka/util/ByteString.scala:158
Taint Flags:

```
155 // override def apply(ignore: TraversableOnce[Byte]): ByteStringBuilder = new ByteStringBuilder
156 // }
157
158 private[akka] object ByteString1C extends Companion {
159 val empty = new ByteString1C(Array.emptyByteArray)
160
161 def fromString(s: String): ByteString1C = new ByteString1C(s.getBytes(StandardCharsets.UTF_8))
```

src/main/scala-2.13/akka/util/ByteString.scala, line 177 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ByteString$ByteString1C
File: src/main/scala-2.13/akka/util/ByteString.scala:177
Taint Flags:

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

Package: akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 177 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
174 * A compact (unsliced) and unfragmented ByteString, implementation of ByteString1C.
175 */
176 @SerialVersionUID(3956956327691936932L)
177 final class ByteString1C private (private val bytes: Array[Byte]) extends CompactByteString {
178 def apply(idx: Int): Byte = bytes(idx)
179
180 override def length: Int = bytes.length
```

src/main/scala-2.13/akka/util/ByteString.scala, line 712 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ByteString$SerializationProxy
File: src/main/scala-2.13/akka/util/ByteString.scala:712
Taint Flags:

```
709 }
710
711 @SerialVersionUID(1L)
712 private class SerializationProxy(@transient private var orig: ByteString) extends Serializable {
713 private def writeObject(out: ObjectOutputStream): Unit = {
714 out.writeByte(orig.byteStringCompanion.SerializationIdentity)
715 orig.writeToOutputStream(out)
```

src/main/scala-2.13/akka/util/ByteString.scala, line 277 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ByteString$ByteString1
File: src/main/scala-2.13/akka/util/ByteString.scala:277
Taint Flags:

```
274 }
275
276 /** INTERNAL API: ByteString backed by exactly one array, with start / end markers */
277 private[akka] object ByteString1 extends Companion {
278 val empty: ByteString1 = new ByteString1(Array.emptyByteArray, 0, 0)
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 277 (Code Correctness: Non-Static Inner Class Implements Serializable) **Low**

```
279 def fromString(s: String): ByteString1 = apply(s.getBytes(StandardCharsets.UTF_8))
280 def apply(bytes: Array[Byte]): ByteString1 = apply(bytes, 0, bytes.length)
```

src/main/scala-2.13/akka/util/ByteString.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: ByteString$ByteString1
File: src/main/scala-2.13/akka/util/ByteString.scala:294
Taint Flags:

```
291 /**
292 * An unfragmented ByteString.
293 */
294 final class ByteString1 private (private val bytes: Array[Byte], private val startIndex: Int, val length: Int)
295 extends ByteString
296 with Serializable {
297
```

Code Correctness: Non-Synchronized Method Overrides Synchronized Method (5 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`:

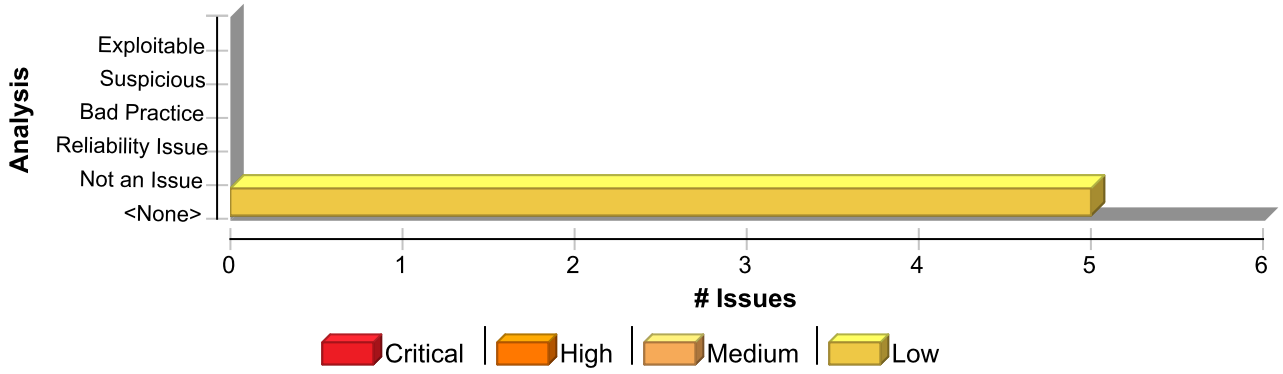
```
public class Bar {
public synchronized void synchronizedMethod() {
    for (int i=0; i<10; i++) System.out.print(i);
    System.out.println();
}
}

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	5	0	0	5
Total	**5**	**0**	**0**	**5**

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

Package: akka.event

src/main/scala/akka/event/EventStream.scala, line 24 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: publish
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:24
Taint Flags:

```
21  * The debug flag in the constructor toggles if operations on this EventStream should also be published
22  * as Debug-Events
23  */
24 class EventStream(sys: ActorSystem, private val debug: Boolean) extends LoggingBus with SubchannelClassification {
25
26  def this(sys: ActorSystem) = this(sys, debug = false)
27
```

Package: akka.event.japi

src/main/scala/akka/event/japi/EventBusJavaAPI.scala, line 99 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: unsubscribe
Enclosing Method: unsubscribe()
File: src/main/scala/akka/event/japi/EventBusJavaAPI.scala:99
Taint Flags:

```
96  * C is the Classifier type
97  */
98 abstract class SubchannelEventBus[E, S, C] extends EventBus[E, S, C] {
99  private val bus = new akka.event.EventBus with akka.event.SubchannelClassification {
100  type Event = E
101  type Subscriber = S
102  type Classifier = C
```

src/main/scala/akka/event/japi/EventBusJavaAPI.scala, line 99 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method) — Low

Issue Details

Kingdom: Code Quality

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

Package: akka.event.japi

src/main/scala/akka/event/japi/EventBusJavaAPI.scala, line 99 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: Function: subscribe
Enclosing Method: subscribe()
File: src/main/scala/akka/event/japi/EventBusJavaAPI.scala:99
Taint Flags:

```
96  * C is the Classifier type
97  */
98 abstract class SubchannelEventBus[E, S, C] extends EventBus[E, S, C] {
99  private val bus = new akka.event.EventBus with akka.event.SubchannelClassification {
100  type Event = E
101  type Subscriber = S
102  type Classifier = C
```

src/main/scala/akka/event/japi/EventBusJavaAPI.scala, line 99 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: publish
Enclosing Method: publish()
File: src/main/scala/akka/event/japi/EventBusJavaAPI.scala:99
Taint Flags:

```
96  * C is the Classifier type
97  */
98 abstract class SubchannelEventBus[E, S, C] extends EventBus[E, S, C] {
99  private val bus = new akka.event.EventBus with akka.event.SubchannelClassification {
100  type Event = E
101  type Subscriber = S
102  type Classifier = C
```

src/main/scala/akka/event/japi/EventBusJavaAPI.scala, line 99 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Non-Synchronized Method Overrides Synchronized Method	Low
Package: akka.event.japi	
src/main/scala/akka/event/japi/EventBusJavaAPI.scala, line 99 (Code Correctness: Non-Synchronized Method Overrides Synchronized Method)	**Low**

Sink: Function: unsubscribe
Enclosing Method: unsubscribe()
File: src/main/scala/akka/event/japi/EventBusJavaAPI.scala:99
Taint Flags:

```
96  * C is the Classifier type
97  */
98 abstract class SubchannelEventBus[E, S, C] extends EventBus[E, S, C] {
99  private val bus = new akka.event.EventBus with akka.event.SubchannelClassification {
100  type Event = E
101  type Subscriber = S
102  type Classifier = C
```

Code Correctness: readObject() Invokes Overridable Function (1 issue)

Abstract

The `readObject()` method within the class calls a function that may be overridden.

Explanation

During deserialization, `readObject()` acts like a constructor, so object initialization is not complete until this function ends. Therefore when a `readObject()` function of a `Serializable` class calls an overridable function, this may provide the overriding method access to the object's state prior to it being fully initialized. **Example 1:** The following `readObject()` function calls a method that can be overridden.

```
  ...
  private void readObject(final ObjectInputStream ois) throws IOException,
ClassNotFoundException {
    checkStream(ois);
    ois.defaultReadObject();
  }

  public void checkStream(ObjectInputStream stream){
    ...
  }
```

Since the function `checkStream()` and its enclosing class are not `final` and public, it means that the function can be overridden, which may mean that an attacker may override the `checkStream()` function in order to get access to the object during deserialization.

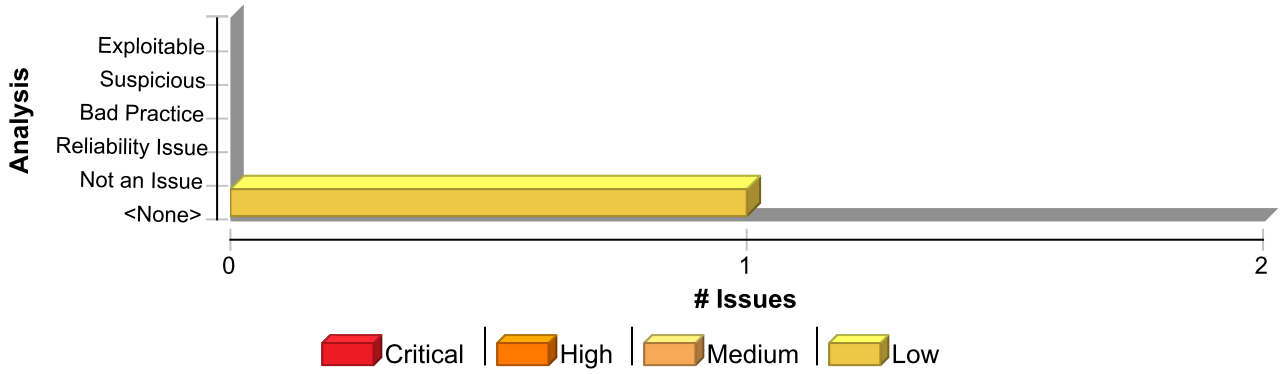
Recommendation

During deserialization, `readObject` 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 `readObject()` function, the `private` access specifier can be used, or the logic could be placed directly into the `readObject()` method itself. **Example 2:** The following prevents the `checkStream()` method from being overridden.

```
  ...
  private void readObject(final ObjectInputStream ois) throws IOException,
ClassNotFoundException {
    checkStream(ois);
    ois.defaultReadObject();
  }

  public final void checkStream(ObjectInputStream stream){
    ...
  }
```

In `Example 2`, `checkStream` was set to `final`, so that it cannot be overridden by a subclass.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: readObject() Invokes Overridable Function	1	0	0	1
Total	**1**	**0**	**0**	**1**

Code Correctness: readObject() Invokes Overridable Function — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 721 (Code Correctness: readObject() Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: orig_=
Enclosing Method: readObject()
File: src/main/scala-2.13/akka/util/ByteString.scala:721
Taint Flags:

```
718 private def readObject(in: ObjectInputStream): Unit = {
719 val serializationId = in.readByte()
720
721 orig = Companion(from = serializationId).readFromInputStream(in)
722 }
723
724 private def readResolve(): AnyRef = orig
```

Dead Code: Expression is Always false (20 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`.

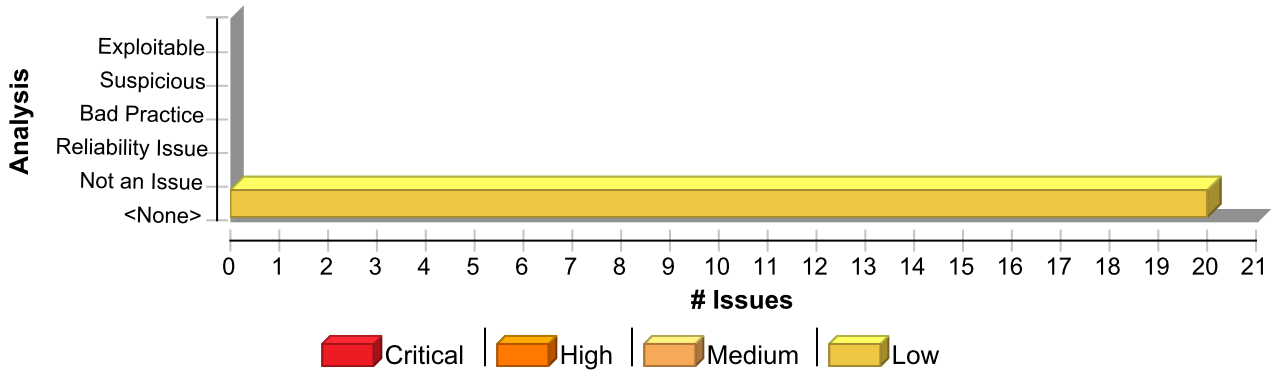
```
public void setUpCalls() {
  boolean firstCall = false;
  boolean secondCall = false;

  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	20	0	0	20
Total	**20**	**0**	**0**	**20**

Dead Code: Expression is Always false — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 129 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: apply()
File: src/main/scala/akka/actor/ActorSystem.scala:129
Taint Flags:

```
126 providerClass match {
127 case "local" => Local
128 // additional fqcn for older configs not using 'remote' or 'cluster'
129 case "remote" | RemoteActorRefProvider => Remote
130 case "cluster" | ClusterActorRefProvider => Cluster
131 case fqcn => Custom(fqcn)
132 }
```

src/main/scala/akka/actor/FSM.scala, line 320 (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.actor

src/main/scala/akka/actor/FSM.scala, line 320 (Dead Code: Expression is Always false) — Low

Sink Details

Sink: IfStatement
Enclosing Method: withNotification()
File: src/main/scala/akka/actor/FSM.scala:320
Taint Flags:

```
317 * INTERNAL API.
318 */
319 private[akka] def withNotification(notifies: Boolean): State[S, D] = {
320 if (notifies)
321 State(stateName, stateData, timeout, stopReason, replies)
322 else
323 new SilentState(stateName, stateData, timeout, stopReason, replies)
```

src/main/scala/akka/actor/FSM.scala, line 984 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: processEvent()
File: src/main/scala/akka/actor/FSM.scala:984
Taint Flags:

```
981 log.debug("processing {} from {} in state {}", event, srcstr, stateName)
982 }
983
984 if (logDepth > 0) {
985 states(pos) = stateName.asInstanceOf[AnyRef]
986 events(pos) = event
987 advance()
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 679 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:679
Taint Flags:

Dead Code: Expression is Always false — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRefProvider.scala, line 679 (Dead Code: Expression is Always false) — Low

```
676 (if (lookupDeploy) deployer.lookup(path) else deploy) match {
677 case Some(d) =>
678 (d.dispatcher, d.mailbox) match {
679 case (Deploy.NoDispatcherGiven, Deploy.NoMailboxGiven) => props
680 case (Deploy.DispatcherSameAsParent, Deploy.NoMailboxGiven) => props.withDispatcher(parentDispatcher)
681 case (dsp, Deploy.NoMailboxGiven) => props.withDispatcher(dsp)
682 case (Deploy.NoDispatcherGiven, mbx) => props.withMailbox(mbx)
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 679 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:679
Taint Flags:

```
676 (if (lookupDeploy) deployer.lookup(path) else deploy) match {
677 case Some(d) =>
678 (d.dispatcher, d.mailbox) match {
679 case (Deploy.NoDispatcherGiven, Deploy.NoMailboxGiven) => props
680 case (Deploy.DispatcherSameAsParent, Deploy.NoMailboxGiven) => props.withDispatcher(parentDispatcher)
681 case (dsp, Deploy.NoMailboxGiven) => props.withDispatcher(dsp)
682 case (Deploy.NoDispatcherGiven, mbx) => props.withMailbox(mbx)
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 680 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:680
Taint Flags:

```
677 case Some(d) =>
678 (d.dispatcher, d.mailbox) match {
679 case (Deploy.NoDispatcherGiven, Deploy.NoMailboxGiven) => props
```

Dead Code: Expression is Always false — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRefProvider.scala, line 680 (Dead Code: Expression is Always false) — Low

```
680 case (Deploy.DispatcherSameAsParent, Deploy.NoMailboxGiven) => props.withDispatcher(parentDispatcher)
681 case (dsp, Deploy.NoMailboxGiven) => props.withDispatcher(dsp)
682 case (Deploy.NoDispatcherGiven, mbx) => props.withMailbox(mbx)
683 case (Deploy.DispatcherSameAsParent, mbx) => props.withDispatcher(parentDispatcher).withMailbox(mbx)
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 681 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:681
Taint Flags:

```
678 (d.dispatcher, d.mailbox) match {
679 case (Deploy.NoDispatcherGiven, Deploy.NoMailboxGiven) => props
680 case (Deploy.DispatcherSameAsParent, Deploy.NoMailboxGiven) => props.withDispatcher(parentDispatcher)
681 case (dsp, Deploy.NoMailboxGiven) => props.withDispatcher(dsp)
682 case (Deploy.NoDispatcherGiven, mbx) => props.withMailbox(mbx)
683 case (Deploy.DispatcherSameAsParent, mbx) => props.withDispatcher(parentDispatcher).withMailbox(mbx)
684 case (dsp, mbx) => props.withDispatcher(dsp).withMailbox(mbx)
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 682 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:682
Taint Flags:

```
679 case (Deploy.NoDispatcherGiven, Deploy.NoMailboxGiven) => props
680 case (Deploy.DispatcherSameAsParent, Deploy.NoMailboxGiven) => props.withDispatcher(parentDispatcher)
681 case (dsp, Deploy.NoMailboxGiven) => props.withDispatcher(dsp)
682 case (Deploy.NoDispatcherGiven, mbx) => props.withMailbox(mbx)
683 case (Deploy.DispatcherSameAsParent, mbx) => props.withDispatcher(parentDispatcher).withMailbox(mbx)
684 case (dsp, mbx) => props.withDispatcher(dsp).withMailbox(mbx)
```

Dead Code: Expression is Always false	Low

Package: akka.actor

src/main/scala/akka/actor/ActorRefProvider.scala, line 682 (Dead Code: Expression is Always false) **Low**

```
685 }
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 680 (Dead Code: Expression is Always false) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:680
Taint Flags:

```
677 case Some(d) =>
678 (d.dispatcher, d.mailbox) match {
679 case (Deploy.NoDispatcherGiven, Deploy.NoMailboxGiven) => props
680 case (Deploy.DispatcherSameAsParent, Deploy.NoMailboxGiven) => props.withDispatcher(parentDispatcher)
681 case (dsp, Deploy.NoMailboxGiven) => props.withDispatcher(dsp)
682 case (Deploy.NoDispatcherGiven, mbx) => props.withMailbox(mbx)
683 case (Deploy.DispatcherSameAsParent, mbx) => props.withDispatcher(parentDispatcher).withMailbox(mbx)
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 683 (Dead Code: Expression is Always false) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:683
Taint Flags:

```
680 case (Deploy.DispatcherSameAsParent, Deploy.NoMailboxGiven) => props.withDispatcher(parentDispatcher)
681 case (dsp, Deploy.NoMailboxGiven) => props.withDispatcher(dsp)
682 case (Deploy.NoDispatcherGiven, mbx) => props.withMailbox(mbx)
683 case (Deploy.DispatcherSameAsParent, mbx) => props.withDispatcher(parentDispatcher).withMailbox(mbx)
684 case (dsp, mbx) => props.withDispatcher(dsp).withMailbox(mbx)
685 }
686 case _ =>
```

Dead Code: Expression is Always false — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 130 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: apply()
File: src/main/scala/akka/actor/ActorSystem.scala:130
Taint Flags:

```
127 case "local" => Local
128 // additional fqcn for older configs not using 'remote' or 'cluster'
129 case "remote" | RemoteActorRefProvider => Remote
130 case "cluster" | ClusterActorRefProvider => Cluster
131 case fqcn => Custom(fqcn)
132 }
133 }
```

src/main/scala/akka/actor/ActorSystem.scala, line 127 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: apply()
File: src/main/scala/akka/actor/ActorSystem.scala:127
Taint Flags:

```
124 /** INTERNAL API */
125 @InternalApi private[akka] def apply(providerClass: String): ProviderSelection =
126 providerClass match {
127 case "local" => Local
128 // additional fqcn for older configs not using 'remote' or 'cluster'
129 case "remote" | RemoteActorRefProvider => Remote
130 case "cluster" | ClusterActorRefProvider => Cluster
```

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsMessage.scala, line 92 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality

Dead Code: Expression is Always false — Low

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsMessage.scala, line 92 (Dead Code: Expression is Always false) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: apply()
File: src/main/scala/akka/io/dns/internal/DnsMessage.scala:92
Taint Flags:

```
89 new MessageFlags(
90 ((if (answer) 0x8000 else 0) |
91 (opCode.id << 11) |
92 (if (authoritativeAnswer) 1 << 10 else 0) |
93 (if (truncated) 1 << 9 else 0) |
94 (if (recursionDesired) 1 << 8 else 0) |
95 (if (recursionAvailable) 1 << 7 else 0) |
```

src/main/scala/akka/io/dns/internal/DnsMessage.scala, line 93 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: apply()
File: src/main/scala/akka/io/dns/internal/DnsMessage.scala:93
Taint Flags:

```
90 ((if (answer) 0x8000 else 0) |
91 (opCode.id << 11) |
92 (if (authoritativeAnswer) 1 << 10 else 0) |
93 (if (truncated) 1 << 9 else 0) |
94 (if (recursionDesired) 1 << 8 else 0) |
95 (if (recursionAvailable) 1 << 7 else 0) |
96 responseCode.id).toShort)
```

src/main/scala/akka/io/dns/internal/DnsMessage.scala, line 90 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Dead Code: Expression is Always false — Low

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsMessage.scala, line 90 (Dead Code: Expression is Always false) — Low

Sink: IfStatement
Enclosing Method: apply()
File: src/main/scala/akka/io/dns/internal/DnsMessage.scala:90
Taint Flags:

```
87 recursionAvailable: Boolean = false,
88 responseCode: ResponseCode.Value = ResponseCode.SUCCESS): MessageFlags = {
89 new MessageFlags(
90 ((if (answer) 0x8000 else 0) |
91 (opCode.id << 11) |
92 (if (authoritativeAnswer) 1 << 10 else 0) |
93 (if (truncated) 1 << 9 else 0) |
```

src/main/scala/akka/io/dns/internal/DnsMessage.scala, line 95 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: apply()
File: src/main/scala/akka/io/dns/internal/DnsMessage.scala:95
Taint Flags:

```
92 (if (authoritativeAnswer) 1 << 10 else 0) |
93 (if (truncated) 1 << 9 else 0) |
94 (if (recursionDesired) 1 << 8 else 0) |
95 (if (recursionAvailable) 1 << 7 else 0) |
96 responseCode.id).toShort)
97 }
98 }
```

Package: akka.util

src/main/scala/akka/util/WildcardIndex.scala, line 15 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: insert()

Dead Code: Expression is Always false — Low

Package: akka.util

src/main/scala/akka/util/WildcardIndex.scala, line 15 (Dead Code: Expression is Always false) — Low

File: src/main/scala/akka/util/WildcardIndex.scala:15
Taint Flags:

```
12 doubleWildcardTree: WildcardTree[T] = WildcardTree[T]()) {
13
14 def insert(elems: Array[String], d: T): WildcardIndex[T] = elems.lastOption match {
15 case Some("**") => copy(doubleWildcardTree = doubleWildcardTree.insert(elems.iterator, d))
16 case Some(_) => copy(wildcardTree = wildcardTree.insert(elems.iterator, d))
17 case _ => this
18 }
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 186 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: drainTo()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:186
Taint Flags:

```
183 if (c eq null) throw new NullPointerException
184 if (c eq this) throw new IllegalArgumentException
185 if (c eq backing) throw new IllegalArgumentException
186 if (maxElements <= 0) 0
187 else {
188 lock.lock()
189 try {
```

src/main/scala-2.13/akka/util/ByteString.scala, line 814 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: grouped()
File: src/main/scala-2.13/akka/util/ByteString.scala:814
Taint Flags:

```
811 override def indexOf[B >: Byte](elem: B, from: Int): Int = indexOf(elem, from)
```

Dead Code: Expression is Always false	Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 814 (Dead Code: Expression is Always false)	Low

```
812
813 override def grouped(size: Int): Iterator[ByteString] = {
814 if (size <= 0) {
815 throw new IllegalArgumentException(s"size=$size must be positive")
816 }
817
```

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/Deployer.scala, line 236 (Dead Code: Expression is Always false)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: apply()
File: src/main/scala/akka/actor/Deployer.scala:236
Taint Flags:

```
233
234 config.root.asScala
235 .flatMap {
236 case ("default", _) => None
237 case (key, value: ConfigObject) => parseConfig(key, value.toConfig)
238 case _ => None
239 }
```

Dead Code: Expression is Always true (9 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`.

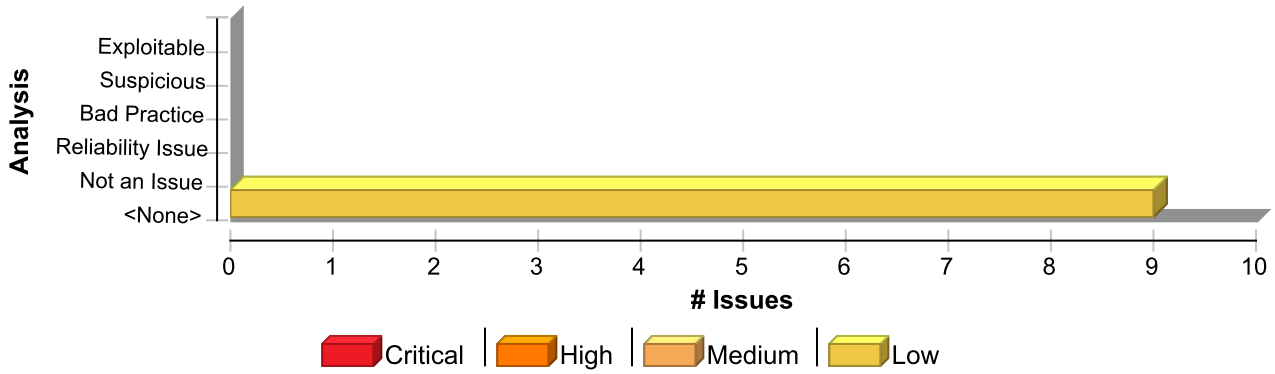
```
public void setUpCalls() {
  boolean firstCall = false;
  boolean secondCall = false;

  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	9	0	0	9
Total	**9**	**0**	**0**	**9**

Dead Code: Expression is Always true — Low

Package: akka.actor

src/main/scala/akka/actor/ActorCell.scala, line 593 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: become()
File: src/main/scala/akka/actor/ActorCell.scala:593
Taint Flags:

```
590 }
591
592 def become(behavior: Actor.Receive, discardOld: Boolean = true): Unit =
593 behaviorStack = behavior :: (if (discardOld && behaviorStack.nonEmpty) behaviorStack.tail else behaviorStack)
594
595 def become(behavior: Procedure[Any]): Unit = become(behavior, discardOld = true)
596
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 676 (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.actor

src/main/scala/akka/actor/ActorRefProvider.scala, line 676 (Dead Code: Expression is Always true) — Low

Sink Details

Sink: IfStatement
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:676
Taint Flags:

```
673
674 val props2 =
675 // mailbox and dispatcher defined in deploy should override props
676 (if (lookupDeploy) deployer.lookup(path) else deploy) match {
677 case Some(d) =>
678 (d.dispatcher, d.mailbox) match {
679 case (Deploy.NoDispatcherGiven, Deploy.NoMailboxGiven) => props
```

src/main/scala/akka/actor/ActorRefProvider.scala, line 712 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: actorOf()
File: src/main/scala/akka/actor/ActorRefProvider.scala:712
Taint Flags:

```
709 }
710
711 case router =>
712 val lookup = if (lookupDeploy) deployer.lookup(path) else None
713 val r = (router :: deploy.map(_.routerConfig).toList ::: lookup.map(_.routerConfig).toList).reduce((a, b) =>
714 b.withFallback(a))
715 val p = props.withRouter(r)
```

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsMessage.scala, line 94 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement

Dead Code: Expression is Always true — Low

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsMessage.scala, line 94 (Dead Code: Expression is Always true) — Low

Enclosing Method: apply()
File: src/main/scala/akka/io/dns/internal/DnsMessage.scala:94
Taint Flags:

```
91  (opCode.id << 11) |
92  (if (authoritativeAnswer) 1 << 10 else 0) |
93  (if (truncated) 1 << 9 else 0) |
94  (if (recursionDesired) 1 << 8 else 0) |
95  (if (recursionAvailable) 1 << 7 else 0) |
96  responseCode.id).toShort)
97  }
```

Package: akka.routing

src/main/scala/akka/routing/RoutedActorCell.scala, line 197 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: src/main/scala/akka/routing/RoutedActorCell.scala:197
Taint Flags:

```
194
195  override def receive =
196  ({
197  case AdjustPoolSize(change: Int) =>
198  if (change > 0) {
199  val newRoutees = Vector.fill(change)(pool.newRoutee(cell.routeeProps, context))
200  cell.addRoutees(newRoutees)
```

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 93 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement

Dead Code: Expression is Always true — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 93 (Dead Code: Expression is Always true) — Low

Enclosing Method: dropWhile()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:93
Taint Flags:

```
90
91 final override def dropWhile(p: Byte => Boolean): this.type = {
92 var stop = false
93 while (!stop && hasNext) {
94 if (p(array(from))) {
95 from = from + 1
96 } else {
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 485 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: indexWhere()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:485
Taint Flags:

```
482 index += 1
483 }
484 var found = false
485 while (!found && hasNext) if (p(next())) {
486 found = true
487 } else {
488 index += 1
```

src/main/scala/akka/util/ManifestInfo.scala, line 180 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: checkSameVersion()
File: src/main/scala/akka/util/ManifestInfo.scala:180
Taint Flags:

Dead Code: Expression is Always true — Low

Package: akka.util

src/main/scala/akka/util/ManifestInfo.scala, line 180 (Dead Code: Expression is Always true) — Low

```
177  throwException: Boolean): Boolean = {
178  ManifestInfo.checkSameVersion(productName, dependencies, versions) match {
179  case Some(message) =>
180  if (logWarning)
181  Logging(system, classOf[ManifestInfo]).warning(message)
182
183  if (throwException)
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 294 (Dead Code: Expression is Always true) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: takeWhile()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:294
Taint Flags:

```
291  final override def takeWhile(p: Byte => Boolean): this.type = {
292  var stop = false
293  val builder = new ListBuffer[ByteArrayIterator]
294  while (!stop && !iterators.isEmpty) {
295  val lastLen = current.len
296  current.takeWhile(p)
297  if (current.hasNext) builder += current
```

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.

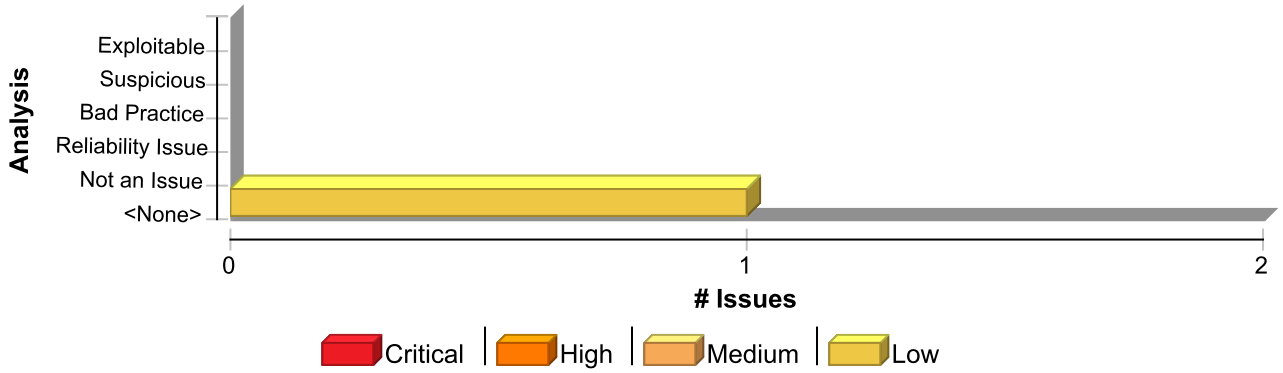
```
  int usrSleepTime = Integer.parseInt(usrInput);
  if (usrSleepTime >= SLEEP_MIN &&
      usrSleepTime <= SLEEP_MAX) {
    Thread.sleep(usrSleepTime);
  } else {
    throw new Exception("Invalid sleep duration");
  }
}
```

Example 4: The following code reads a String from a zip file just as in `Example 2`, except 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.util

src/main/scala-2.13/akka/util/ByteString.scala, line 168 (Denial of Service) — Low

Issue Details

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

Source Details

Source: java.io.ObjectInputStream.readInt()
From: akka.util.ByteString$ByteString1C.readFromInputStream
File: src/main/scala-2.13/akka/util/ByteString.scala:166

```
163  val SerializationIdentity = 1.toByte
164
165  def readFromInputStream(is: ObjectInputStream): ByteString1C = {
166  val length = is.readInt()
167  val arr = new Array[Byte](length)
168  is.readFully(arr, 0, length)
169  ByteString1C(arr)
```

Sink Details

Sink: java.io.ObjectInputStream.readFully()
Enclosing Method: readFromInputStream()
File: src/main/scala-2.13/akka/util/ByteString.scala:168
Taint Flags: NUMBER, SERIALIZED

Denial of Service	Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteString.scala, line 168 (Denial of Service)	Low

```
165 def readFromInputStream(is: ObjectInputStream): ByteString1C = {
166 val length = is.readInt()
167 val arr = new Array[Byte](length)
168 is.readFully(arr, 0, length)
169 ByteString1C(arr)
170 }
171 }
```

Insecure Randomness (9 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.

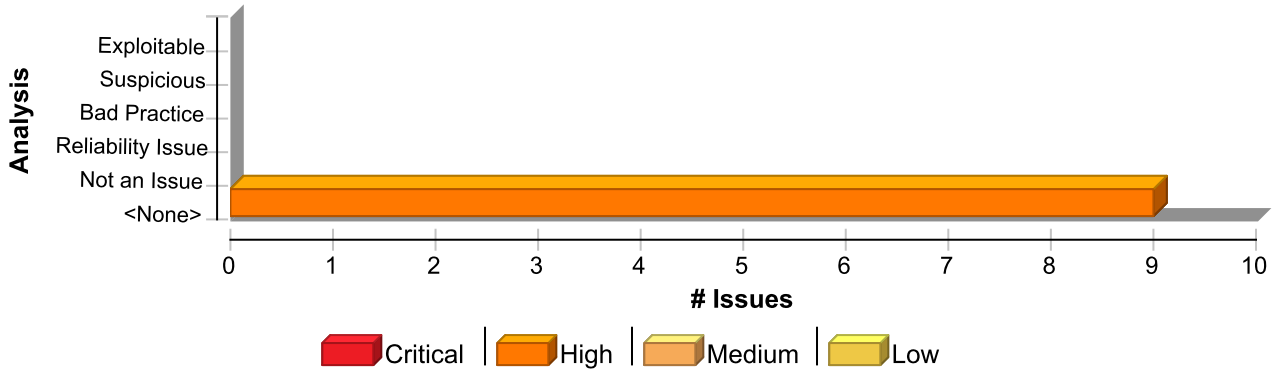
```
String GenerateReceiptURL(String baseUrl) {
    Random ranGen = new Random();
    ranGen.setSeed((new Date()).getTime());
    return (baseUrl + ranGen.nextInt(400000000) + ".html");
}
```

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	9	0	0	9
Total	**9**	**0**	**0**	**9**

Insecure Randomness — High

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 808 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextLong()
Enclosing Method: ActorSystemImpl()
File: src/main/scala/akka/actor/ActorSystem.scala:808
Taint Flags:

```
805 setup: ActorSystemSetup)
806 extends ExtendedActorSystem {
807
808 val uid: Long = ThreadLocalRandom.current.nextLong()
809
810 if (!name.matches("""^[a-zA-Z0-9][a-zA-Z0-9-_]*$"""))
811 throw new IllegalArgumentException(
```

src/main/scala/akka/actor/ActorCell.scala, line 386 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: newUid()
File: src/main/scala/akka/actor/ActorCell.scala:386
Taint Flags:

Insecure Randomness — High

Package: akka.actor

src/main/scala/akka/actor/ActorCell.scala, line 386 (Insecure Randomness) — High

```
383 @tailrec final def newUid(): Int = {
384 // Note that this uid is also used as hashCode in ActorRef, so be careful
385 // to not break hashing if you change the way uid is generated
386 val uid = ThreadLocalRandom.current.nextInt()
387 if (uid == undefinedUid) newUid()
388 else uid
389 }
```

Package: akka.pattern

src/main/scala/akka/pattern/CircuitBreaker.scala, line 1045 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextDouble()
Enclosing Method: _enter()
File: src/main/scala/akka/pattern/CircuitBreaker.scala:1045
Taint Flags:

```
1042 scheduler.scheduleOnce(currentResetTimeout) {
1043 attemptReset()
1044 }
1045 val rnd = 1.0 + ThreadLocalRandom.current().nextDouble() * randomFactor
1046 val nextResetTimeout = currentResetTimeout * exponentialBackoffFactor * rnd match {
1047 case f: FiniteDuration => f
1048 case _ => currentResetTimeout
```

src/main/scala/akka/pattern/BackoffSupervisor.scala, line 317 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextDouble()
Enclosing Method: calculateDelay()
File: src/main/scala/akka/pattern/BackoffSupervisor.scala:317
Taint Flags:

```
314 minBackoff: FiniteDuration,
315 maxBackoff: FiniteDuration,
316 randomFactor: Double): FiniteDuration = {
317 val rnd = 1.0 + ThreadLocalRandom.current().nextDouble() * randomFactor
318 val calculatedDuration = Try(maxBackoff.min(minBackoff * math.pow(2, restartCount)) * rnd).getOrElse(maxBackoff)
```

Insecure Randomness — High

Package: akka.pattern

src/main/scala/akka/pattern/BackoffSupervisor.scala, line 317 (Insecure Randomness) — High

```
319 calculatedDuration match {
320 case f: FiniteDuration => f
```

Package: akka.routing

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 299 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: explore()
File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:299
Taint Flags:

```
296 }
297
298 private def explore(currentSize: PoolSize): Int = {
299 val change = Math.max(1, random.nextInt(Math.ceil(currentSize * exploreStepSize).toInt))
300 if (random.nextDouble() < chanceOfScalingDownWhenFull)
301 -change
302 else
```

src/main/scala/akka/routing/SmallestMailbox.scala, line 64 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: selectNext()
File: src/main/scala/akka/routing/SmallestMailbox.scala:64
Taint Flags:

```
61 NoRoutee
62 else if (at >= targets.size) {
63 if (deep) {
64 if (isTerminated(proposedTarget)) targets(ThreadLocalRandom.current.nextInt(targets.size)) else proposedTarget
65 } else selectNext(targets, proposedTarget, currentScore, 0, deep = true)
66 } else {
67 val target = targets(at)
```

Insecure Randomness — High

Package: akka.routing

src/main/scala/akka/routing/Random.scala, line 31 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: select()
File: src/main/scala/akka/routing/Random.scala:31
Taint Flags:

```
28 final class RandomRoutingLogic extends RoutingLogic {
29  override def select(message: Any, routees: immutable.IndexedSeq[Routee]): Routee =
30  if (routees.isEmpty) NoRoutee
31  else routees(ThreadLocalRandom.current.nextInt(routees.size))
32 }
33
34 /**
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 268 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextDouble()
Enclosing Method: resize()
File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:268
Taint Flags:

```
265  } else if (performanceLog.isEmpty || record.underutilizationStreak.isDefined) {
266  0
267  } else {
268  if (!stopExploring && random.nextDouble() < explorationProbability)
269  explore(currentSize)
270  else
271  optimize(currentSize)
```

src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 300 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Insecure Randomness	High
Package: akka.routing	
src/main/scala/akka/routing/OptimalSizeExploringResizer.scala, line 300 (Insecure Randomness)	**High**

Sink Details

Sink: nextDouble()
Enclosing Method: explore()
File: src/main/scala/akka/routing/OptimalSizeExploringResizer.scala:300
Taint Flags:

```
297
298 private def explore(currentSize: PoolSize): Int = {
299 val change = Math.max(1, random.nextInt(Math.ceil(currentSize * exploreStepSize).toInt))
300 if (random.nextDouble() < chanceOfScalingDownWhenFull)
301 -change
302 else
303 change
```

J2EE Bad Practices: JVM Termination (4 issues)

Abstract

A web application should not attempt to shut down its container.

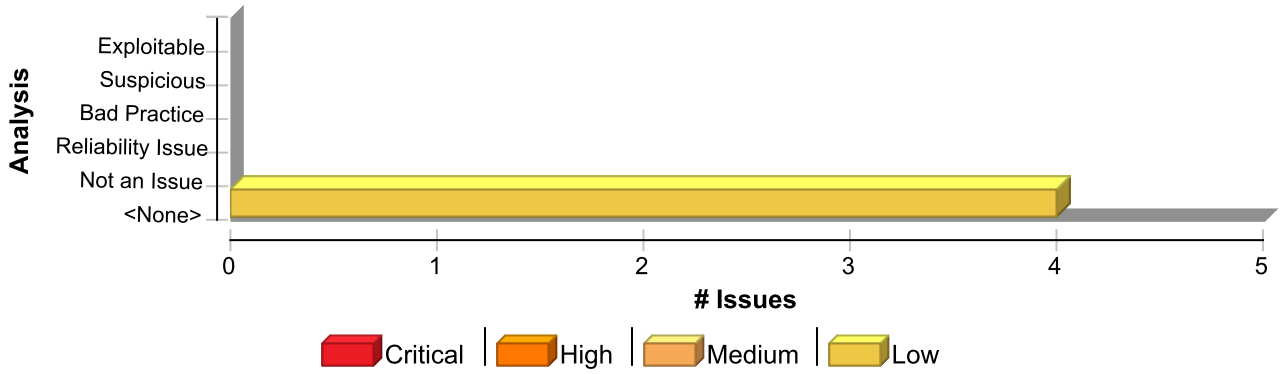
Explanation

It is never a good idea for a web application to attempt to shut down the application container. A call to a termination method is probably part of leftover debug code or code imported from a non-J2EE application.

Recommendation

Never call a termination method within a web application. Such method calls in a J2EE application indicates poor software hygiene and should be removed. Regardless of whether there is a perceived threat, it is unlikely that there is a legitimate reason for such code to remain in the application.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: JVM Termination	4	0	0	4
Total	**4**	**0**	**0**	**4**

J2EE Bad Practices: JVM Termination	Low
Package: akka.actor	
src/main/scala/akka/actor/ActorSystem.scala, line 846 (J2EE Bad Practices: JVM Termination)	**Low**

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: exit()
Enclosing Method: uncaughtException()
File: src/main/scala/akka/actor/ActorSystem.scala:846
Taint Flags:

J2EE Bad Practices: JVM Termination — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 846 (J2EE Bad Practices: JVM Termination) — Low

```
843
844 if (settings.JvmExitOnFatalError)
845 try logFatalError("shutting down JVM since 'akka.jvm-exit-on-fatal-error' is enabled for", cause, thread)
846 finally System.exit(-1)
847 else
848 try logFatalError("shutting down", cause, thread)
849 finally terminate()
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 248 (J2EE Bad Practices: JVM Termination) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: exit()
Enclosing Method: run()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:248
Taint Flags:

```
245 val t = new Thread {
246 override def run(): Unit = {
247 if (Try(Await.ready(system.whenTerminated, timeout)).isFailure && !runningJvmHook)
248 System.exit(exitCode)
249 }
250 }
251 t.setName("CoordinatedShutdown-exit")
```

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 258 (J2EE Bad Practices: JVM Termination) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: exit()
Enclosing Method: apply()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:258
Taint Flags:

```
255 if (terminateActorSystem) {
256 system.finalTerminate()
```

J2EE Bad Practices: JVM Termination	Low

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 258 (J2EE Bad Practices: JVM Termination) **Low**

```
257 system.whenTerminated.map { _ =>
258 if (exitJvm && !runningJvmHook) System.exit(exitCode)
259 Done
260 }(ExecutionContexts.parasitic)
261 } else if (exitJvm) {
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 262 (J2EE Bad Practices: JVM Termination) **Low**

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: exit()
Enclosing Method: apply()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:262
Taint Flags:

```
259 Done
260 }(ExecutionContexts.parasitic)
261 } else if (exitJvm) {
262 System.exit(exitCode)
263 Future.successful(Done)
264 } else
265 Future.successful(Done)
```

J2EE Bad Practices: Leftover Debug Code (1 issue)

Abstract

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

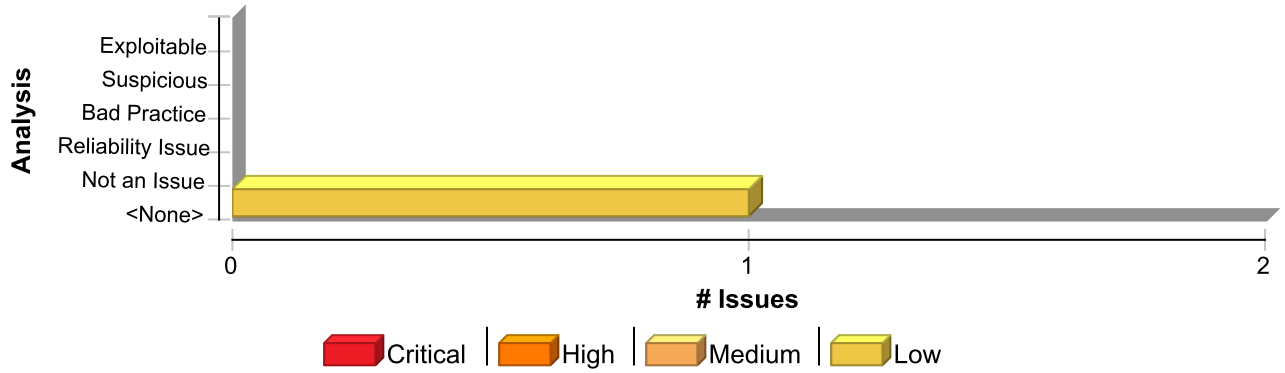
Explanation

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

Recommendation

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

Issue Summary



Engine Breakdown

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

J2EE Bad Practices: Leftover Debug Code	Low
Package: akka	
src/main/scala/akka/Main.scala, line 28 (J2EE Bad Practices: Leftover Debug Code)	**Low**

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

J2EE Bad Practices: Leftover Debug Code	Low

Package: akka

src/main/scala/akka/Main.scala, line 28 (J2EE Bad Practices: Leftover Debug Code) Low

Sink: Function: main
Enclosing Method: main()
File: src/main/scala/akka/Main.scala:28
Taint Flags:

```
25 /**
26 * @param args one argument: the class of the application supervisor actor
27 */
28 def main(args: Array[String]): Unit = {
29 if (args.length != 1) {
30 println("you need to provide exactly one argument: the class of the application supervisor actor")
31 } else {
```

J2EE Bad Practices: Sockets (8 issues)

Abstract

Socket-based communication in web applications is prone to error.

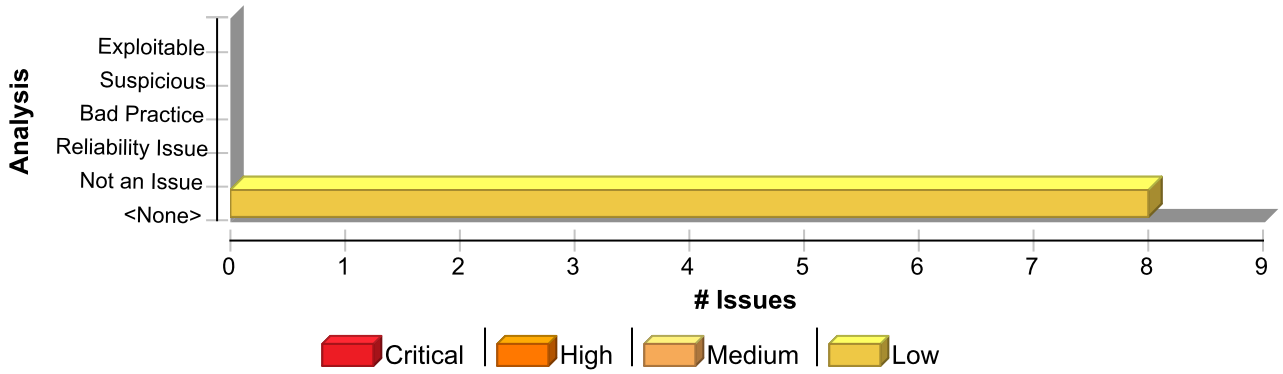
Explanation

The J2EE standard permits the use of sockets only for the purpose of communication with legacy systems when no higher-level protocol is available. Authoring your own communication protocol requires wrestling with difficult security issues, including: - In-band versus out-of-band signaling - Compatibility between protocol versions - Channel security - Error handling - Network constraints (firewalls) - Session management Without significant scrutiny by a security expert, chances are good that a custom communication protocol will suffer from security problems. Many of the same issues apply to a custom implementation of a standard protocol. While there are usually more resources available that address security concerns related to implementing a standard protocol, these resources are also available to attackers.

Recommendation

Replace a custom communication protocol with an industry standard protocol or framework. Consider whether you can use a protocol such as HTTP, FTP, SMTP, CORBA, RMI/IIOP, EJB, or SOAP. Consider the security track record of the protocol implementation you choose.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: Sockets	8	0	0	8
Total	**8**	**0**	**0**	**8**

J2EE Bad Practices: Sockets	Low
Package: akka.io	
src/main/scala/akka/io/WithUdpSend.scala, line 52 (J2EE Bad Practices: Sockets)	**Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

J2EE Bad Practices: Sockets — Low

Package: akka.io

src/main/scala/akka/io/WithUdpSend.scala, line 52 (J2EE Bad Practices: Sockets) — Low

Sink: InetSocketAddress()
Enclosing Method: applyOrElse()
File: src/main/scala/akka/io/WithUdpSend.scala:52
Taint Flags:

```
49 Dns.resolve(DnsProtocol.Resolve(send.target.getHostName), context.system, self) match {
50 case Some(r) =>
51 try {
52 pendingSend = pendingSend.copy(target = new InetSocketAddress(r.address(), pendingSend.target.getPort))
53 doSend(registration)
54 } catch {
55 case NonFatal(e) =>
```

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 142 (J2EE Bad Practices: Sockets) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: InetSocketAddress()
Enclosing Method: parseNameserverAddress()
File: src/main/scala/akka/io/dns/DnsSettings.scala:142
Taint Flags:

```
139 @InternalApi private[akka] def parseNameserverAddress(str: String): InetSocketAddress =
140 str match {
141 case inetSocketAddress(host, port) =>
142 new InetSocketAddress(host, Option(port).fold(DnsFallbackPort)(_.toInt))
143 case unexpected =>
144 throw new IllegalArgumentException(s"Unparseable address string: $unexpected") // will not happen, for exhaustiveness check
145 }
```

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsClient.scala, line 53 (J2EE Bad Practices: Sockets) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: InetSocketAddress()
Enclosing Method: preStart()
File: src/main/scala/akka/io/dns/internal/DnsClient.scala:53
Taint Flags:

J2EE Bad Practices: Sockets — Low

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/DnsClient.scala, line 53 (J2EE Bad Practices: Sockets) — Low

```
50 lazy val tcpDnsClient: ActorRef = createTcpClient()
51
52 override def preStart() = {
53 udp ! Udp.Bind(self, new InetSocketAddress(InetAddress.getByAddress(Array.ofDim(4)), 0))
54 }
55
56 def receive: Receive = {
```

Package: src.main.scala.akka.io

src/main/scala/akka/io/UdpConnection.scala, line 63 (J2EE Bad Practices: Sockets) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: InetSocketAddress()
Enclosing Method: apply()
File: src/main/scala/akka/io/UdpConnection.scala:63
Taint Flags:

```
60 def resolving(): Receive = {
61 case r: DnsProtocol.Resolved =>
62 reportConnectFailure {
63 doConnect(new InetSocketAddress(r.address(), remoteAddress.getPort))
64 }
65 case Failure(ex) =>
66 // async-dns responds with a Failure on DNS server lookup failure
```

src/main/scala/akka/io/TcpOutgoingConnection.scala, line 84 (J2EE Bad Practices: Sockets) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: InetSocketAddress()
Enclosing Method: apply()
File: src/main/scala/akka/io/TcpOutgoingConnection.scala:84
Taint Flags:

```
81 def resolving(registration: ChannelRegistration): Receive = {
82 case resolved: DnsProtocol.Resolved =>
83 reportConnectFailure {
84 register(new InetSocketAddress(resolved.address(), remoteAddress.getPort), registration)
```

J2EE Bad Practices: Sockets — Low

Package: src.main.scala.akka.io

src/main/scala/akka/io/TcpOutgoingConnection.scala, line 84 (J2EE Bad Practices: Sockets) — Low

```
85  }
86  case ReceiveTimeout =>
87  connectionTimeout()
```

src/main/scala/akka/io/UdpConnection.scala, line 49 (J2EE Bad Practices: Sockets) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: InetSocketAddress()
Enclosing Method: apply()
File: src/main/scala/akka/io/UdpConnection.scala:49
Taint Flags:

```
46  Dns.resolve(DnsProtocol.Resolve(remoteAddress.getHostName), context.system, self) match {
47  case Some(r) =>
48  reportConnectFailure {
49  doConnect(new InetSocketAddress(r.address(), remoteAddress.getPort))
50  }
51  case None =>
52  context.become(resolving())
```

src/main/scala/akka/io/TcpOutgoingConnection.scala, line 71 (J2EE Bad Practices: Sockets) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: InetSocketAddress()
Enclosing Method: apply()
File: src/main/scala/akka/io/TcpOutgoingConnection.scala:71
Taint Flags:

```
68  case None =>
69  context.become(resolving(registration))
70  case Some(resolved) =>
71  register(new InetSocketAddress(resolved.address(), remoteAddress.getPort), registration)
72  }
73  } else {
74  register(remoteAddress, registration)
```

J2EE Bad Practices: Sockets	Low

Package: src.main.scala.akka.io.dns

src/main/scala/akka/io/dns/DnsSettings.scala, line 165 (J2EE Bad Practices: Sockets)	Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: InetSocketAddress()
Enclosing Method: apply()
File: src/main/scala/akka/io/dns/DnsSettings.scala:165
Taint Flags:

```
162 case -1 => DnsFallbackPort
163 case selected => selected
164 }
165 new InetSocketAddress(host, port)
166 }
167 }
168
```

J2EE Bad Practices: Threads (58 issues)

Abstract

Thread management in a web application is forbidden in some circumstances and is always highly error prone.

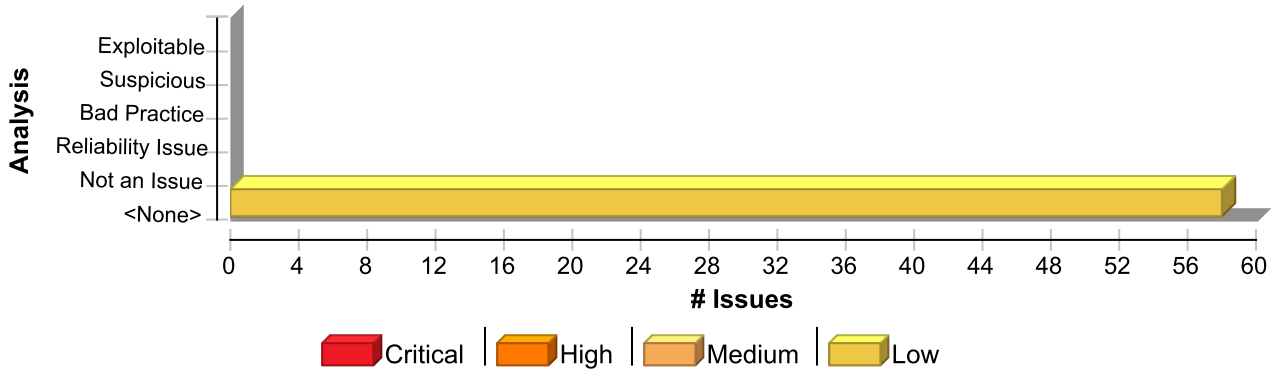
Explanation

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	58	0	0	58
Total	**58**	**0**	**0**	**58**

J2EE Bad Practices: Threads	**Low**
Package: akka.actor	
src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 276 (J2EE Bad Practices: Threads)	**Low**

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:276

J2EE Bad Practices: Threads — Low

Package: akka.actor

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 276 (J2EE Bad Practices: Threads) — Low

Taint Flags:

```
273 log.error(t, "exception on LARS' timer thread")
274 stopped.get match {
275 case null =>
276 val thread = threadFactory.newThread(this)
277 log.info("starting new LARS thread")
278 try thread.start()
279 catch {
```

src/main/scala/akka/actor/TypedActor.scala, line 215 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: ThreadLocal()
Enclosing Method: TypedActor()
File: src/main/scala/akka/actor/TypedActor.scala:215
Taint Flags:

```
212 }
213
214 private val selfReference = new ThreadLocal[AnyRef]
215 private val currentContext = new ThreadLocal[ActorContext]
216
217 @SerialVersionUID(1L)
218 private case object NullResponse
```

src/main/scala/akka/actor/TypedActor.scala, line 214 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: ThreadLocal()
Enclosing Method: TypedActor()
File: src/main/scala/akka/actor/TypedActor.scala:214
Taint Flags:

```
211 }
212 }
213
214 private val selfReference = new ThreadLocal[AnyRef]
```

J2EE Bad Practices: Threads — Low

Package: akka.actor

src/main/scala/akka/actor/TypedActor.scala, line 214 (J2EE Bad Practices: Threads) — Low

```
215 private val currentContext = new ThreadLocal[ActorContext]
216
217 @SerialVersionUID(1L)
```

src/main/scala/akka/actor/ActorCell.scala, line 366 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: ThreadLocal()
Enclosing Method: ActorCell$$anon$1()
File: src/main/scala/akka/actor/ActorCell.scala:366
Taint Flags:

```
363 * for! (waves hand)
364 */
365 private[akka] object ActorCell {
366 val contextStack = new ThreadLocal[List[ActorContext]] {
367 override def initialValue: List[ActorContext] = Nil
368 }
369
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 335 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: start()
Enclosing Method: LightArrayRevolverScheduler()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:335
Taint Flags:

```
332 }
333 })
334
335 timerThread.start()
336 }
337
338 object LightArrayRevolverScheduler {
```

J2EE Bad Practices: Threads — Low

Package: akka.actor

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 90 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: sleep()
Enclosing Method: waitNanos()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:90
Taint Flags:

```
87 protected def waitNanos(nanos: Long): Unit = {
88 // see https://www.javamex.com/tutorials/threads/sleep_issues.shtml
89 val sleepMs = if (Helpers.isWindows) (nanos + 4999999) / 10000000 * 10 else (nanos + 999999) / 1000000
90 try Thread.sleep(sleepMs)
91 catch {
92 case _: InterruptedException => Thread.currentThread().interrupt() // we got woken up
93 }
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 245 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: Thread()
Enclosing Method: CoordinatedShutdown$$anon$1()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:245
Taint Flags:

```
242 // We must spawn a separate thread to not block current thread,
243 // since that would have blocked the shutdown of the ActorSystem.
244 val timeout = coord.timeout(PhaseActorSystemTerminate)
245 val t = new Thread {
246 override def run(): Unit = {
247 if (Try(Await.ready(system.whenTerminated, timeout)).isFailure && !runningJvmHook)
248 System.exit(exitCode)
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 838 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

J2EE Bad Practices: Threads — Low

Package: akka.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 838 (J2EE Bad Practices: Threads) — Low

Sink Details

Sink: Thread()
Enclosing Method: CoordinatedShutdown$$anon$3()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:838
Taint Flags:

```
835 val currentLatch = _jvmHooksLatch.get
836 val newLatch = new CountDownLatch(currentLatch.getCount.toInt + 1)
837 if (_jvmHooksLatch.compareAndSet(currentLatch, newLatch)) {
838 val thread = new Thread {
839 override def run(): Unit = {
840 try hook
841 finally _jvmHooksLatch.get.countDown()
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 113 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:113
Taint Flags:

```
110 new AtomicLong(clock() + initialDelay.toNanos) with Runnable {
111 override def run(): Unit = {
112 try {
113 runnable.run()
114 val driftNanos = clock() - getAndAdd(delay.toNanos)
115 if (self.get != null)
116 swap(schedule(executor, this, Duration.fromNanos(Math.max(delay.toNanos - driftNanos, 1))))
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 278 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: start()
Enclosing Method: run()

J2EE Bad Practices: Threads — Low

Package: akka.actor

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 278 (J2EE Bad Practices: Threads) — Low

File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:278
Taint Flags:

```
275 case null =>
276 val thread = threadFactory.newThread(this)
277 log.info("starting new LARS thread")
278 try thread.start()
279 catch {
280 case e: Throwable =>
281 log.error(e, "LARS cannot start new thread, ship’s going down!")
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 373 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: removeShutdownHook()
Enclosing Method: removeHook()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:373
Taint Flags:

```
370 @InternalApi
371 private[akka] object JVMShutdownHooks extends JVMShutdownHooks {
372  override def addHook(t: Thread): Unit = Runtime.getRuntime.addShutdownHook(t)
373  override def removeHook(t: Thread): Boolean = Runtime.getRuntime.removeShutdownHook(t)
374 }
375
376 final class CoordinatedShutdown private[akka] (
```

src/main/scala/akka/actor/Scheduler.scala, line 83 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/actor/Scheduler.scala:83
Taint Flags:

```
80 new Runnable {
81 override def run(): Unit = {
```

J2EE Bad Practices: Threads — Low

Package: akka.actor

src/main/scala/akka/actor/Scheduler.scala, line 83 (J2EE Bad Practices: Threads) — Low

```
82 try {
83 runnable.run()
84 if (self.get != null)
85 swap(scheduleOnce(delay, this))
86 } catch {
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 372 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: addShutdownHook()
Enclosing Method: addHook()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:372
Taint Flags:

```
369 */
370 @InternalApi
371 private[akka] object JVMShutdownHooks extends JVMShutdownHooks {
372 override def addHook(t: Thread): Unit = Runtime.getRuntime.addShutdownHook(t)
373 override def removeHook(t: Thread): Boolean = Runtime.getRuntime.removeShutdownHook(t)
374 }
375
```

src/main/scala/akka/actor/ActorSystem.scala, line 1285 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: applyOrElse()
File: src/main/scala/akka/actor/ActorSystem.scala:1285
Taint Flags:

```
1282 final def add(r: Runnable): Unit = {
1283 @tailrec def addRec(r: Runnable, p: Promise[T]): Unit = ref.get match {
1284 case null => throw new RejectedExecutionException("ActorSystem already terminated.")
1285 case some if ref.compareAndSet(some, p) => some.completeWith(p.future.andThen { case _ => r.run() })
1286 case _ => addRec(r, p)
1287 }
1288 addRec(r, Promise[T]())
```

J2EE Bad Practices: Threads — Low

Package: akka.actor

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 374 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:374
Taint Flags:

```
371 }
372
373 // This should only be called in execDirectly
374 override def run(): Unit = extractTask(ExecutedTask).run()
375
376 override def cancel(): Boolean = extractTask(CancelledTask) match {
377 case ExecutedTask | CancelledTask => false
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 230 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: LightArrayRevolverScheduler()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:230
Taint Flags:

```
227 } else Future.successful(Nil)
228 }
229
230 @volatile private var timerThread: Thread = threadFactory.newThread(new Runnable {
231
232 var tick = startTick
233 var totalTick: Long = tick // tick count that doesn't wrap around, used for calculating sleep time
```

src/main/scala/akka/actor/ActorCell.scala, line 651 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

J2EE Bad Practices: Threads — Low

Package: akka.actor

src/main/scala/akka/actor/ActorCell.scala, line 651 (J2EE Bad Practices: Threads) — Low

Sink Details

Sink: interrupt()
Enclosing Method: create()
File: src/main/scala/akka/actor/ActorCell.scala:651
Taint Flags:

```
648 } catch {
649 case e: InterruptedException =>
650 failActor()
651 Thread.currentThread().interrupt()
652 throw ActorInitializationException(self, "interruption during creation", e)
653 case NonFatal(e) =>
654 failActor()
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 92 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: interrupt()
Enclosing Method: waitNanos()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:92
Taint Flags:

```
89 val sleepMs = if (Helpers.isWindows) (nanos + 4999999) / 10000000 * 10 else (nanos + 999999) / 1000000
90 try Thread.sleep(sleepMs)
91 catch {
92 case _: InterruptedException => Thread.currentThread().interrupt() // we got woken up
93 }
94 }
95
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 368 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: interrupt()
Enclosing Method: executeTask()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:368

J2EE Bad Practices: Threads — Low

Package: akka.actor

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 368 (J2EE Bad Practices: Threads) — Low

Taint Flags:

```
365 executionContext.execute(other)
366 true
367 } catch {
368 case _: InterruptedException => Thread.currentThread().interrupt(); false
369 case NonFatal(e) => executionContext.reportFailure(e); false
370 }
371 }
```

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 160 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: runTask()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:160
Taint Flags:

```
157 override def close(): Unit = {
158
159 def runTask(task: Runnable): Unit = {
160 try task.run()
161 catch {
162 case e: InterruptedException => throw e
163 case _: SchedulerException => // ignore terminated actors
```

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/FaultHandling.scala, line 337 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: interrupt()
Enclosing Method: applyOrElse()
File: src/main/scala/akka/actor/dungeon/FaultHandling.scala:337
Taint Flags:

J2EE Bad Practices: Threads — Low

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/FaultHandling.scala, line 337 (J2EE Bad Practices: Threads) — Low

```
334 final protected def handleNonFatalOrInterruptedException(thunk: (Throwable) => Unit): Catcher[Unit] = {
335 case e: InterruptedException =>
336 thunk(e)
337 Thread.currentThread().interrupt()
338 case NonFatal(e) =>
339 thunk(e)
340 }
```

src/main/scala/akka/actor/dungeon/Dispatch.scala, line 128 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: interrupt()
Enclosing Method: applyOrElse()
File: src/main/scala/akka/actor/dungeon/Dispatch.scala:128
Taint Flags:

```
125 private def handleException: Catcher[Unit] = {
126 case e: InterruptedException =>
127 system.eventStream.publish(Error(e, self.path.toString, clazz(actor), "interrupted during message send"))
128 Thread.currentThread().interrupt()
129 case NonFatal(e) =>
130 val message = e match {
131 case n: NoStackTrace => "swallowing exception during message send: " + n.getMessage
```

Package: akka.dispatch

src/main/scala/akka/dispatch/ThreadPoolBuilder.scala, line 38 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: availableProcessors()
Enclosing Method: scaledPoolSize()
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:38
Taint Flags:

```
35 val defaultRejectionPolicy: RejectedExecutionHandler = new SaneRejectedExecutionHandler()
36
37 def scaledPoolSize(floor: Int, multiplier: Double, ceiling: Int): Int =
```

J2EE Bad Practices: Threads — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/ThreadPoolBuilder.scala, line 38 (J2EE Bad Practices: Threads) — Low

```
38 math.min(math.max((Runtime.getRuntime.availableProcessors * multiplier).ceil.toInt, floor), ceiling)
39
40 def arrayBlockingQueue(capacity: Int, fair: Boolean): QueueFactory =
41 () => new ArrayBlockingQueue[Runnable](capacity, fair)
```

src/main/scala/akka/dispatch/ThreadPoolBuilder.scala, line 200 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: newThread()
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:200
Taint Flags:

```
197 t
198 }
199
200 def newThread(runnable: Runnable): Thread = wire(new Thread(runnable, name + "-" + counter.incrementAndGet()))
201
202 def withName(newName: String): MonitorableThreadFactory = copy(newName)
203
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 49 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:49
Taint Flags:

```
46 final override def isBatchable: Boolean = akka.dispatch.internal.ScalaBatchable.isBatchable(runnable)
47
48 def run(): Unit =
49 try runnable.run()
50 catch {
51 case NonFatal(e) => eventStream.publish(Error(e, "TaskInvocation", this.getClass, e.getMessage))
```

J2EE Bad Practices: Threads	Low

Package: akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 49 (J2EE Bad Practices: Threads) — Low

```
52 } finally cleanup()
```

src/main/scala/akka/dispatch/BatchingExecutor.scala, line 90 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: ThreadLocal()
Enclosing Method: BatchingExecutor()
File: src/main/scala/akka/dispatch/BatchingExecutor.scala:90
Taint Flags:

```
87 }
88 }
89
90 private[this] val _blockContext = new ThreadLocal[BlockContext]()
91
92 private[this] final class BlockableBatch extends AbstractBatch with BlockContext {
93 // this method runs in the delegate ExecutionContext's thread
```

src/main/scala/akka/dispatch/BatchingExecutor.scala, line 58 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: ThreadLocal()
Enclosing Method: BatchingExecutor()
File: src/main/scala/akka/dispatch/BatchingExecutor.scala:58
Taint Flags:

```
55 private[akka] trait BatchingExecutor extends Executor {
56
57 // invariant: if "_tasksLocal.get ne null" then we are inside Batch.run; if it is null, we are outside
58 private[this] val _tasksLocal = new ThreadLocal[AbstractBatch]()
59
60 private[this] abstract class AbstractBatch extends ArrayDeque[Runnable](4) with Runnable {
61 @tailrec final def processBatch(batch: AbstractBatch): Unit =
```

J2EE Bad Practices: Threads — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Dispatcher.scala, line 132 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: registerForExecution()
File: src/main/scala/akka/dispatch/Dispatcher.scala:132
Taint Flags:

```
129 } catch {
130 case _: RejectedExecutionException =>
131 try {
132 executorService.execute(mbox)
133 true
134 } catch { //Retry once
135 case e: RejectedExecutionException =>
```

src/main/scala/akka/dispatch/Dispatcher.scala, line 127 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: registerForExecution()
File: src/main/scala/akka/dispatch/Dispatcher.scala:127
Taint Flags:

```
124 if (mbox.canBeScheduledForExecution(hasMessageHint, hasSystemMessageHint)) { //This needs to be here to ensure thread safety and no races
125 if (mbox.setAsScheduled()) {
126 try {
127 executorService.execute(mbox)
128 true
129 } catch {
130 case _: RejectedExecutionException =>
```

src/main/scala/akka/dispatch/BatchingExecutor.scala, line 63 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

J2EE Bad Practices: Threads — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/BatchingExecutor.scala, line 63 (J2EE Bad Practices: Threads) — Low

Sink Details

Sink: run()
Enclosing Method: processBatch()
File: src/main/scala/akka/dispatch/BatchingExecutor.scala:63
Taint Flags:

```
60 private[this] abstract class AbstractBatch extends ArrayDeque[Runnable](4) with Runnable {
61 @tailrec final def processBatch(batch: AbstractBatch): Unit =
62 if ((batch eq this) && !batch.isEmpty) {
63 batch.poll().run()
64 processBatch(_tasksLocal.get) // If this is null, then we have been using managed blocking, so bail out
65 }
66
```

src/main/scala/akka/dispatch/Dispatcher.scala, line 43 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: execute()
File: src/main/scala/akka/dispatch/Dispatcher.scala:43
Taint Flags:

```
40
41 import configurator.prerequisites._
42
43 private class LazyExecutorServiceDelegate(factory: ExecutorServiceFactory) extends ExecutorServiceDelegate {
44 lazy val executor: ExecutorService = factory.createExecutorService
45 def copy(): LazyExecutorServiceDelegate = new LazyExecutorServiceDelegate(factory)
46 }
```

src/main/scala/akka/dispatch/Mailbox.scala, line 246 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: interrupt()
Enclosing Method: exec()
File: src/main/scala/akka/dispatch/Mailbox.scala:246
Taint Flags:

J2EE Bad Practices: Threads — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 246 (J2EE Bad Practices: Threads) — Low

```
243 run(); false
244 } catch {
245 case _: InterruptedException =>
246 Thread.currentThread().interrupt()
247 false
248 case anything: Throwable =>
249 val t = Thread.currentThread()
```

src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala, line 48 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: exec()
File: src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala:48
Taint Flags:

```
45 override def setRawResult(unit: Unit): Unit = ()
46 override def exec(): Boolean =
47 try {
48 runnable.run(); true
49 } catch {
50 case _: InterruptedException =>
51 Thread.currentThread.interrupt()
```

src/main/scala/akka/dispatch/Dispatcher.scala, line 85 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: executeTask()
File: src/main/scala/akka/dispatch/Dispatcher.scala:85
Taint Flags:

```
82 } catch {
83 case e: RejectedExecutionException =>
84 try {
85 executorService.execute(invocation)
86 } catch {
```

J2EE Bad Practices: Threads — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Dispatcher.scala, line 85 (J2EE Bad Practices: Threads) — Low

```
87 case e2: RejectedExecutionException =>
88 eventStream.publish(Error(e, getClass.getName, getClass, "executeTask was rejected twice!"))
```

src/main/scala/akka/dispatch/Dispatcher.scala, line 81 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: executeTask()
File: src/main/scala/akka/dispatch/Dispatcher.scala:81
Taint Flags:

```
78 */
79 protected[akka] def executeTask(invocation: TaskInvocation): Unit = {
80 try {
81 executorService.execute(invocation)
82 } catch {
83 case e: RejectedExecutionException =>
84 try {
```

src/main/scala/akka/dispatch/Future.scala, line 99 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: execute()
File: src/main/scala/akka/dispatch/Future.scala:99
Taint Flags:

```
96 */
97 @InternalApi
98 @deprecated("Use ExecutionContexts.parasitic instead", "2.6.4")
99 private[akka] object sameThreadExecutionContext extends ExecutionContext with BatchingExecutor {
100 override protected def unbatchedExecute(runnable: Runnable): Unit = parasitic.execute(runnable)
101 override protected def resubmitOnBlock: Boolean = false // No point since we execute on same thread
102 override def reportFailure(t: Throwable): Unit =
```

src/main/scala/akka/dispatch/Mailbox.scala, line 243 (J2EE Bad Practices: Threads) — Low

Issue Details

J2EE Bad Practices: Threads — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 243 (J2EE Bad Practices: Threads) — Low

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: exec()
File: src/main/scala/akka/dispatch/Mailbox.scala:243
Taint Flags:

```
240 override final def setRawResult(unit: Unit): Unit = ()
241 final override def exec(): Boolean =
242 try {
243 run(); false
244 } catch {
245 case _: InterruptedException =>
246 Thread.currentThread().interrupt()
```

src/main/scala/akka/dispatch/ThreadPoolBuilder.scala, line 219 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: execute()
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:219
Taint Flags:

```
216
217 def executor: ExecutorService
218
219 def execute(command: Runnable) = executor.execute(command)
220
221 def shutdown(): Unit = { executor.shutdown() }
222
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 248 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

J2EE Bad Practices: Threads — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 248 (J2EE Bad Practices: Threads) — Low

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:248
Taint Flags:

```
245 }
246 case RESCHEDULED =>
247 if (updateShutdownSchedule(RESCHEDULED, SCHEDULED)) scheduleShutdownAction()
248 else run()
249 case UNSCHEDULED =>
250 case unexpected =>
251 throw new IllegalArgumentException(s"Unexpected actor class marker: $unexpected") // will not happen, for exhaustiveness check
```

src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala, line 51 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: interrupt()
Enclosing Method: exec()
File: src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala:51
Taint Flags:

```
48 runnable.run(); true
49 } catch {
50 case _: InterruptedException =>
51 Thread.currentThread.interrupt()
52 false
53 case anything: Throwable =>
54 val t = Thread.currentThread()
```

src/main/scala/akka/dispatch/ThreadPoolBuilder.scala, line 235 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: submit()
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:235
Taint Flags:

J2EE Bad Practices: Threads — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/ThreadPoolBuilder.scala, line 235 (J2EE Bad Practices: Threads) — Low

```
232
233  def submit[T](runnable: Runnable, t: T) = executor.submit(runnable, t)
234
235  def submit(runnable: Runnable) = executor.submit(runnable)
236
237  def invokeAll[T](callables: Collection[_ <: Callable[T]]) = executor.invokeAll(callables)
238
```

src/main/scala/akka/dispatch/ThreadPoolBuilder.scala, line 233 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: submit()
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:233
Taint Flags:

```
230
231  def submit[T](callable: Callable[T]) = executor.submit(callable)
232
233  def submit[T](runnable: Runnable, t: T) = executor.submit(runnable, t)
234
235  def submit(runnable: Runnable) = executor.submit(runnable)
236
```

src/main/scala/akka/dispatch/ThreadPoolBuilder.scala, line 256 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: rejectedExecution()
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:256
Taint Flags:

```
253 class SaneRejectedExecutionHandler extends RejectedExecutionHandler {
254  def rejectedExecution(runnable: Runnable, threadPoolExecutor: ThreadPoolExecutor): Unit = {
255  if (threadPoolExecutor.isShutdown) throw new RejectedExecutionException("Shutdown")
```

J2EE Bad Practices: Threads — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/ThreadPoolBuilder.scala, line 256 (J2EE Bad Practices: Threads) — Low

```
256  else runnable.run()
257  }
258 }
259
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 197 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: execute()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:197
Taint Flags:

```
194  private def scheduleShutdownAction(): Unit = {
195  // IllegalStateException is thrown if scheduler has been shutdown
196  try prerequisites.scheduler.scheduleOnce(shutdownTimeout, shutdownAction)(new ExecutionContext {
197  override def execute(runnable: Runnable): Unit = runnable.run()
198  override def reportFailure(t: Throwable): Unit = MessageDispatcher.this.reportFailure(t)
199  })
200  catch {
```

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 96 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: execute()
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:96
Taint Flags:

```
93  }
94 }
95
96 abstract class MessageDispatcher(val configurator: MessageDispatcherConfigurator)
97  extends AbstractMessageDispatcher
98  with BatchingExecutor
```

J2EE Bad Practices: Threads	Low

Package: akka.dispatch

src/main/scala/akka/dispatch/AbstractDispatcher.scala, line 96 (J2EE Bad Practices: Threads) — Low

```
99  with ExecutionContextExecutor {
```

Package: akka.dispatch.affinity

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 253 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: executeNext()
File: src/main/scala/akka/dispatch/affinity/AffinityPool.scala:253
Taint Flags:

```
250  val c = q.poll()
251  val next = c ne null
252  if (next) {
253  c.run()
254  idleStrategy.reset()
255  } else {
256  idleStrategy.idle() // if not wait for a bit
```

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 240 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: AffinityPool$AffinityPoolWorker()
File: src/main/scala/akka/dispatch/affinity/AffinityPool.scala:240
Taint Flags:

```
237
238  private[this] final class AffinityPoolWorker(val q: BoundedAffinityTaskQueue, val idleStrategy: IdleStrategy)
239  extends Runnable {
240  val thread: Thread = threadFactory.newThread(this)
241
242  def start(): Unit =
243  if (thread eq null)
```

J2EE Bad Practices: Threads — Low

Package: akka.dispatch.affinity

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 245 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: start()
Enclosing Method: start()
File: src/main/scala/akka/dispatch/affinity/AffinityPool.scala:245
Taint Flags:

```
242 def start(): Unit =
243 if (thread eq null)
244 throw new IllegalStateException(s"Was not able to allocate worker thread for ${AffinityPool.this}")
245 else thread.start()
246
247 override def run(): Unit = {
248 // Returns true if it executed something, false otherwise
```

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 289 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: interrupt()
Enclosing Method: stop()
File: src/main/scala/akka/dispatch/affinity/AffinityPool.scala:289
Taint Flags:

```
286 }
287 }
288
289 def stop(): Unit = if (!thread.isInterrupted) thread.interrupt()
290
291 def stopIfIdle(): Unit = if (idleStrategy.isIdling) stop()
292 }
```

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 87 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

J2EE Bad Practices: Threads — Low

Package: akka.dispatch.affinity

src/main/scala/akka/dispatch/affinity/AffinityPool.scala, line 87 (J2EE Bad Practices: Threads) — Low

Sink Details

Sink: yield()
Enclosing Method: idle()
File: src/main/scala/akka/dispatch/affinity/AffinityPool.scala:87
Taint Flags:

```
84 if (turns > maxYields) {
85 parkPeriodNs = minParkPeriodNs
86 transitionTo(Parking)
87 } else Thread.`yield`()
88 case Parking =>
89 LockSupport.parkNanos(parkPeriodNs)
90 parkPeriodNs = Math.min(parkPeriodNs << 1, maxParkPeriodNs)
```

Package: akka.io

src/main/scala/akka/io/TcpConnection.scala, line 516 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: doWrite()
File: src/main/scala/akka/io/TcpConnection.scala:516
Taint Flags:

```
513 with Runnable {
514
515 def doWrite(info: ConnectionInfo): PendingWrite = {
516 tcp.fileIoDispatcher.execute(this)
517 this
518 }
519
```

src/main/scala/akka/io/SelectionHandler.scala, line 185 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/io/SelectionHandler.scala:185

J2EE Bad Practices: Threads — Low

Package: akka.io

src/main/scala/akka/io/SelectionHandler.scala, line 185 (J2EE Bad Practices: Threads) — Low

Taint Flags:

```
182
183 override def run(): Unit =
184 if (selector.isOpen)
185 try super.run()
186 finally executionContext.execute(this) // re-schedule select behind all currently queued tasks
187 }
188
```

Package: akka.util

src/main/scala/akka/util/SerializedSuspendableExecutionContext.scala, line 74 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/util/SerializedSuspendableExecutionContext.scala:74
Taint Flags:

```
71 poll() match {
72 case null => ()
73 case some =>
74 try some.run()
75 catch { case NonFatal(t) => context.reportFailure(t) }
76 run(done + 1)
77 }
```

src/main/scala/akka/util/SerializedSuspendableExecutionContext.scala, line 76 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/util/SerializedSuspendableExecutionContext.scala:76
Taint Flags:

```
73 case some =>
```

J2EE Bad Practices: Threads — Low

Package: akka.util

src/main/scala/akka/util/SerializedSuspendableExecutionContext.scala, line 76 (J2EE Bad Practices: Threads) — Low

```
74 try some.run()
75 catch { case NonFatal(t) => context.reportFailure(t) }
76 run(done + 1)
77 }
78 }
79 try run(0)
```

src/main/scala/akka/util/SerializedSuspendableExecutionContext.scala, line 79 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: run()
File: src/main/scala/akka/util/SerializedSuspendableExecutionContext.scala:79
Taint Flags:

```
76 run(done + 1)
77 }
78 }
79 try run(0)
80 finally remState(On)
81 }
82
```

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 252 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: start()
Enclosing Method: apply()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:252
Taint Flags:

```
249 }
250 }
251 t.setName("CoordinatedShutdown-exit")
```

J2EE Bad Practices: Threads — Low

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 252 (J2EE Bad Practices: Threads) — Low

```
252 t.start()
253 }
254
255 if (terminateActorSystem) {
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 902 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: apply()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:902
Taint Flags:

```
899 * shutdown hooks the standard library JVM shutdown hooks APIs are better suited.
900 */
901 def addCancellableJvmShutdownHook(hook: Runnable): Cancellable =
902 addCancellableJvmShutdownHook(hook.run())
903
904 }
905
```

src/main/scala/akka/actor/CoordinatedShutdown.scala, line 887 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: apply()
File: src/main/scala/akka/actor/CoordinatedShutdown.scala:887
Taint Flags:

```
884 * hooks, e.g. those shutting down Artery.
885 */
886 def addJvmShutdownHook(hook: Runnable): Unit =
887 addJvmShutdownHook(hook.run())
888
889 /**
```

J2EE Bad Practices: Threads	Low
Package: src.main.scala.akka.actor	
src/main/scala/akka/actor/CoordinatedShutdown.scala, line 887 (J2EE Bad Practices: Threads)	**Low**

890 * Java API: Add a JVM shutdown hook that will be run when the JVM process

Key Management: Hardcoded Encryption Key (2 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:

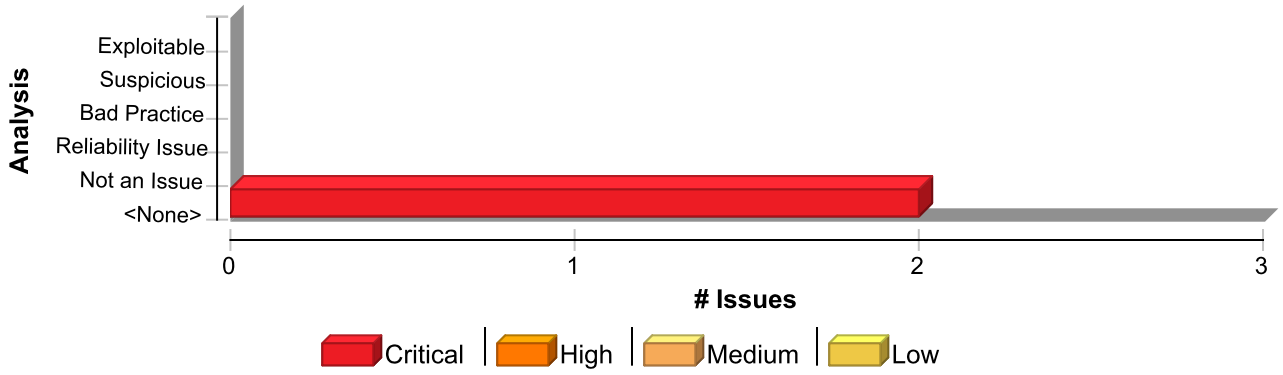
```
...
private static final String encryptionKey = "lakdsljkalkjlksdfkl";
byte[] keyBytes = encryptionKey.getBytes();
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher encryptCipher = Cipher.getInstance("AES");
encryptCipher.init(Cipher.ENCRYPT_MODE, key);
...
```

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

Key Management: Hardcoded Encryption Key — Critical

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 438 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: key
Enclosing Method: ActorSystem$Settings()
File: src/main/scala/akka/actor/ActorSystem.scala:438
Taint Flags:

```
435 }
436 final val LogDeadLettersDuringShutdown: Boolean = config.getBoolean("akka.log-dead-letters-during-shutdown")
437 final val LogDeadLettersSuspendDuration: Duration = {
438 val key = "akka.log-dead-letters-suspend-duration"
439 toRootLowerCase(config.getString(key)) match {
440 case "infinite" => Duration.Inf
441 case _ => config.getMillisDuration(key)
```

src/main/scala/akka/actor/ActorSystem.scala, line 438 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: key
Enclosing Method: ActorSystem$Settings()
File: src/main/scala/akka/actor/ActorSystem.scala:438
Taint Flags:

```
435 }
436 final val LogDeadLettersDuringShutdown: Boolean = config.getBoolean("akka.log-dead-letters-during-shutdown")
437 final val LogDeadLettersSuspendDuration: Duration = {
438 val key = "akka.log-dead-letters-suspend-duration"
439 toRootLowerCase(config.getString(key)) match {
440 case "infinite" => Duration.Inf
441 case _ => config.getMillisDuration(key)
```

Missing Check against Null (4 issues)

Abstract

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

Explanation

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

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

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

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

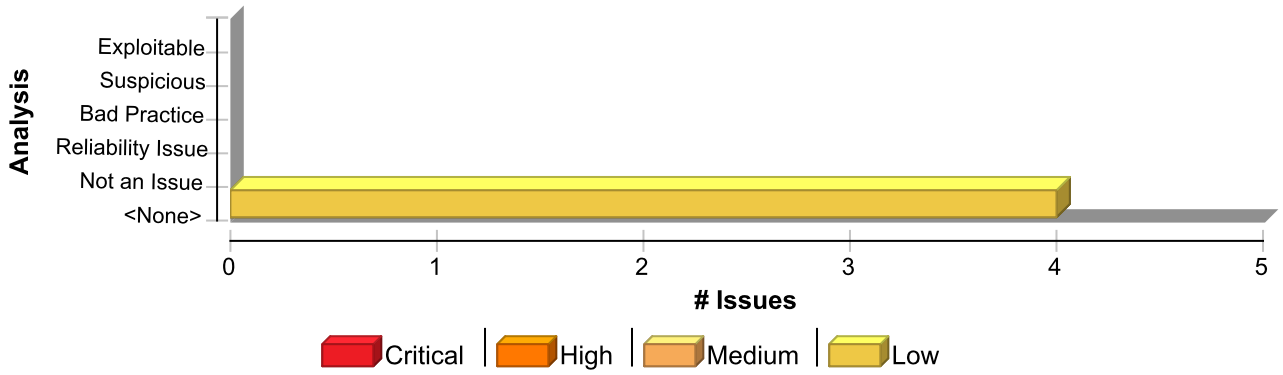
The traditional defense of this coding error is: "I know the requested value will always exist because.... If it does not exist, the program cannot perform the desired behavior so it doesn't matter whether I handle the error or simply allow the program to die dereferencing a `null` value." But attackers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

Recommendation

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

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

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Missing Check against Null	4	0	0	4
Total	**4**	**0**	**0**	**4**

Missing Check against Null — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 1466 (Missing Check against Null) — Low

Issue Details

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

Sink Details

Sink: getComponentType() : Class.getComponentType may return NULL
Enclosing Method: format1()
File: src/main/scala/akka/event/Logging.scala:1466
Taint Flags:

```
1463  * there are more than four arguments.
1464  */
1465  private def format1(t: String, arg: Any): String = arg match {
1466  case a: Array[_] if !a.getClass.getComponentType.isPrimitive => formatImpl(t, a.toSeq)
1467  case a: Array[_] => formatImpl(t, a.map(_.asInstanceOf[AnyRef]).toSeq)
1468  case x => format(t, x)
1469  }
```

src/main/scala/akka/event/Logging.scala, line 1946 (Missing Check against Null) — Low

Issue Details

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

Sink Details

Sink: getComponentType() : Class.getComponentType may return NULL

Missing Check against Null — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 1946 (Missing Check against Null) — Low

Enclosing Method: format1()
File: src/main/scala/akka/event/Logging.scala:1946
Taint Flags:

```
1943
1944  // Copy of LoggingAdapter.format1 due to binary compatibility restrictions
1945  private def format1(t: String, arg: Any): String = arg match {
1946  case a: Array[_] if !a.getClass.getComponentType.isPrimitive => format(t, a.toIndexedSeq)
1947  case a: Array[_] => format(t, a.map(_.asInstanceOf[AnyRef]).toIndexedSeq)
1948  case x => format(t, x)
1949  }
```

Package: akka.util

src/main/scala/akka/util/LineNumbers.scala, line 192 (Missing Check against Null) — Low

Issue Details

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

Sink Details

Sink: cl = getClassLoader() : Class.getClassLoader may return NULL
Enclosing Method: getStreamForClass()
File: src/main/scala/akka/util/LineNumbers.scala:192
Taint Flags:

```
189
190  private def getStreamForClass(c: Class[_]): Option[(InputStream, None.type)] = {
191  val resource = c.getName.replace('.', '/') + ".class"
192  val cl = c.getClassLoader
193  val r = cl.getResourceAsStream(resource)
194  if (debug) println(s"LNB: resource '$resource' resolved to stream $r")
195  Option(r).map(_ -> None)
```

src/main/scala/akka/util/LineNumbers.scala, line 204 (Missing Check against Null) — Low

Issue Details

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

Sink Details

Sink: getClassLoader() : Class.getClassLoader may return NULL
Enclosing Method: getStreamForLambda()
File: src/main/scala/akka/util/LineNumbers.scala:204
Taint Flags:

```
201  val writeReplace = c.getDeclaredMethod("writeReplace")
```

Missing Check against Null	Low

Package: akka.util

src/main/scala/akka/util/LineNumbers.scala, line 204 (Missing Check against Null)	Low

```
202 writeReplace.setAccessible(true)
203 writeReplace.invoke(l) match {
204 case serialized: SerializedLambda =>
205 if (debug)
206 println(s"LNB: found Lambda implemented in ${serialized.getImplClass}:${serialized.getImplMethodName}")
207 Option(c.getClassLoader.getResourceAsStream(serialized.getImplClass + ".class"))
```

Null Dereference (3 issues)

Abstract

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

Explanation

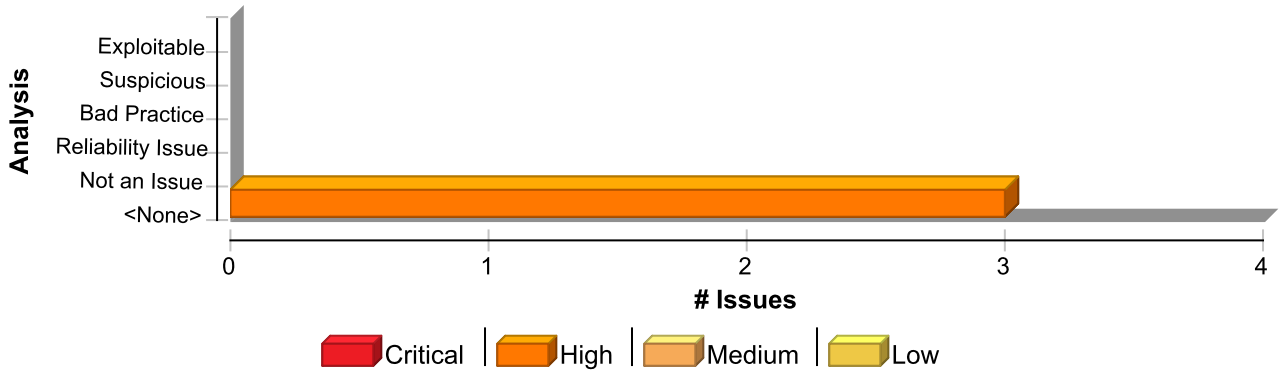
Null-pointer exceptions usually occur when one or more of the programmer's assumptions is violated. A dereference-after-store error occurs when a program explicitly sets an object to `null` and dereferences it later. This error is often the result of a programmer initializing a variable to `null` when it is declared. Most null-pointer issues result in general software reliability problems, but if attackers can intentionally trigger a null-pointer dereference, they can use the resulting exception to bypass security logic or to cause the application to reveal debugging information that will be valuable in planning subsequent attacks. **Example:** In the following code, the programmer explicitly sets the variable `foo` to `null`. Later, the programmer dereferences `foo` before checking the object for a `null` value.

```
Foo foo = null;
...
foo.setBar(val);
...
}
```

Recommendation

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

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Null Dereference	3	0	0	3
Total	**3**	**0**	**0**	**3**

Null Dereference	**High**
Package: akka.dispatch	
src/main/scala/akka/dispatch/Mailbox.scala, line 313 (Null Dereference)	**High**
Issue Details	

Null Dereference — High

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 313 (Null Dereference) — High

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

Sink Details

Sink: Dereferenced : dlm
Enclosing Method: processAllSystemMessages()
File: src/main/scala/akka/dispatch/Mailbox.scala:313
Taint Flags:

```
310 val msg = messageList.head
311 messageList = messageList.tail
312 msg.unlink()
313 try dlm.systemEnqueue(actor.self, msg)
314 catch {
315 case e: InterruptedException => interruption = e
316 case NonFatal(e) =>
```

src/main/scala/akka/dispatch/Mailbox.scala, line 326 (Null Dereference) — High

Issue Details

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

Sink Details

Sink: Dereferenced : interruption
Enclosing Method: processAllSystemMessages()
File: src/main/scala/akka/dispatch/Mailbox.scala:326
Taint Flags:

```
323 }
324 }
325 // if we got an interrupted exception while handling system messages, then rethrow it
326 if (interruption ne null) {
327 Thread.interrupted() // clear interrupted flag before throwing according to java convention
328 throw interruption
329 }
```

Package: akka.serialization

src/main/scala/akka/serialization/Serialization.scala, line 72 (Null Dereference) — High

Issue Details

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

Sink Details

Null Dereference	High
Package: akka.serialization	
src/main/scala/akka/serialization/Serialization.scala, line 72 (Null Dereference)	**High**

Sink: Dereferenced : originalSystem
Enclosing Method: serializedActorPath()
File: src/main/scala/akka/serialization/Serialization.scala:72
Taint Flags:

```
69 originalSystem match {
70 case null => path.toSerializationFormat
71 case system =>
72 try path.toSerializationFormatWithAddress(system.provider.getDefaultAddress)
73 catch { case NonFatal(_) => path.toSerializationFormat }
74 }
75 case Information(address, system) =>
```

Object Model Violation: Erroneous clone() Method (3 issues)

Abstract

A `clone()` method should call `super.clone()` to obtain the new object.

Explanation

All implementations of `clone()` should obtain the new object by calling `super.clone()`. If a class fails to follow this convention, a subclass's `clone()` method will return an object of the wrong type. **Example 1:** The following two classes demonstrate a bug introduced by failing to call `super.clone()`. Because of the way `Kibitzer` implements `clone()`, `FancyKibitzer`'s clone method will return an object of type `Kibitzer` instead of `FancyKibitzer`.

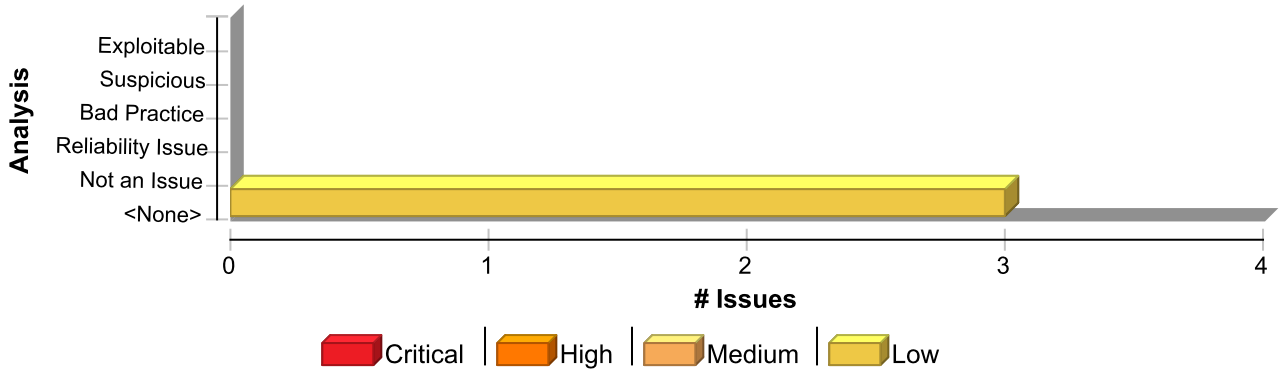
```
public class Kibitzer implements Cloneable {
  public Object clone() throws CloneNotSupportedException {
    Object returnMe = new Kibitzer();
    ...
  }
}

public class FancyKibitzer extends Kibitzer
                           implements Cloneable {
  public Object clone() throws CloneNotSupportedException {
    Object returnMe = super.clone();
    ...
  }
}
```

Recommendation

Always obtain the new object by calling `super.clone()`. The java.lang.Object implementation of `clone()` will always return an object of the correct type. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
public class Kibitzer implements Cloneable {
  public Object clone() throws CloneNotSupportedException {
    Object returnMe = super.clone();
    ...
  }
}

public class FancyKibitzer extends Kibitzer
                           implements Cloneable {
  public Object clone() throws CloneNotSupportedException {
    Object returnMe = super.clone();
    ...
  }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Object Model Violation: Erroneous clone() Method	3	0	0	3
Total	**3**	**0**	**0**	**3**

Object Model Violation: Erroneous clone() Method — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 259 (Object Model Violation: Erroneous clone() Method) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: clone
Enclosing Method: clone()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:259
Taint Flags:

```
256 case io => super.++(io)
257 }
258
259 final override def clone: MultiByteArrayIterator = {
260 val clonedIterators: List[ByteArrayIterator] = iterators.iterator.map(_.clone).to(List)
261 new MultiByteArrayIterator(clonedIterators)
262 }
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 437 (Object Model Violation: Erroneous clone() Method) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: clone

Object Model Violation: Erroneous clone() Method — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 437 (Object Model Violation: Erroneous clone() Method) — Low

Enclosing Method: clone()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:437
Taint Flags:

```
434 // *must* be overridden by derived classes. This construction is necessary
435 // to specialize the return type, as the method is already implemented in
436 // the parent class.
437 override def clone: ByteIterator =
438 throw new UnsupportedOperationException("Method clone is not implemented in ByteIterator")
439
440 override def duplicate: (ByteIterator, ByteIterator) = (this, clone)
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 72 (Object Model Violation: Erroneous clone() Method) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: clone
Enclosing Method: clone()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:72
Taint Flags:

```
69 case io => super.++(io)
70 }
71
72 final override def clone: ByteArrayIterator = new ByteArrayIterator(array, from, until)
73
74 final override def take(n: Int): this.type = {
75 if (n < len) until = { if (n > 0) (from + n) else from }
```

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()`.

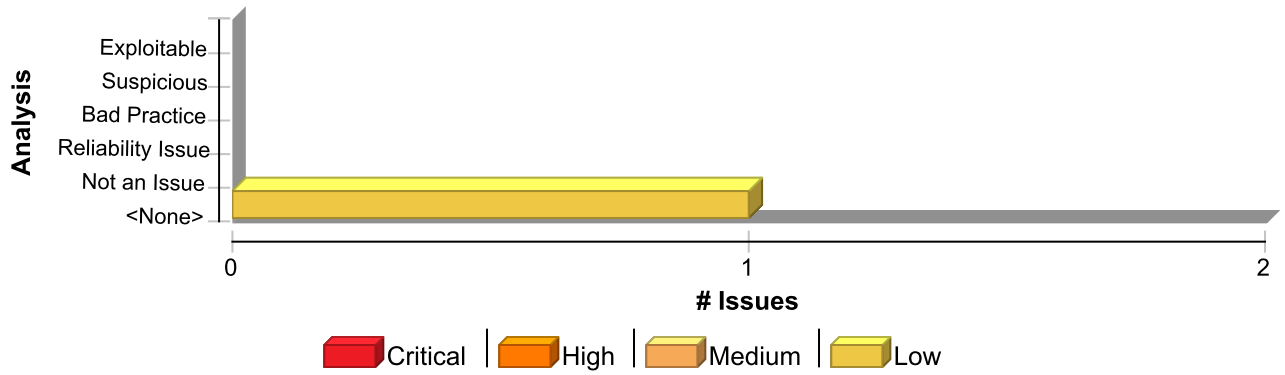
```
public class halfway() {
  public boolean equals(Object obj) {
    ...
  }
}
```

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.actor

src/main/scala/akka/actor/FSM.scala, line 227 (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: src/main/scala/akka/actor/FSM.scala:227
Taint Flags:

```
224 }
225 }
226
227 override def equals(that: Any) = that match {
228 case other: State[_, _] =>
229 other.canEqual(this) &&
230 this.stateName == other.stateName &&
```

Often Misused: Authentication (9 issues)

Abstract

Attackers may spoof DNS entries. Do not rely on DNS names for security.

Explanation

Many DNS servers are susceptible to spoofing attacks, so you should assume that your software will someday run in an environment with a compromised DNS server. If attackers are allowed to make DNS updates (sometimes called DNS cache poisoning), they can route your network traffic through their machines or make it appear as if their IP addresses are part of your domain. Do not base the security of your system on DNS names. **Example:** The following code uses a DNS lookup to determine whether an inbound request is from a trusted host. If an attacker can poison the DNS cache, they can gain trusted status.

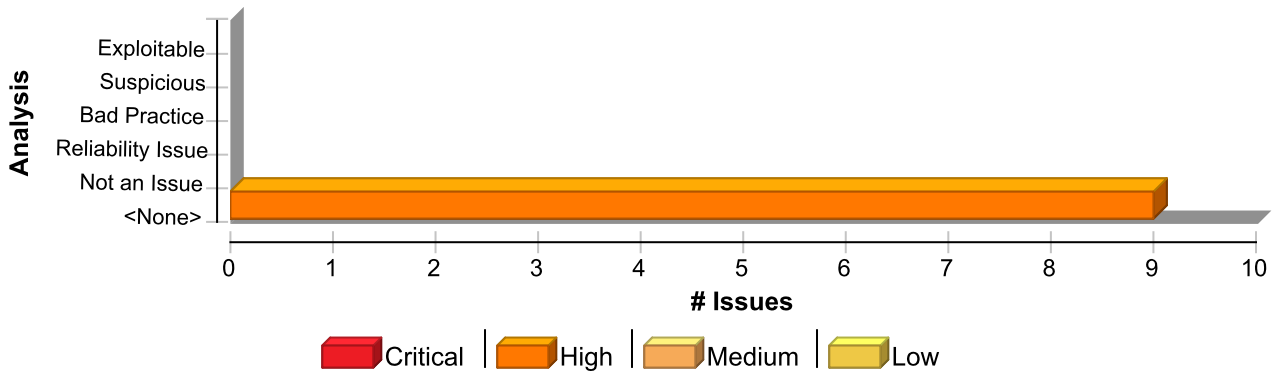
```
String ip = request.getRemoteAddr();
InetAddress addr = InetAddress.getByName(ip);
if (addr.getCanonicalHostName().endsWith("trustme.com")) {
trusted = true;
}
```

IP addresses are more reliable than DNS names, but they can also be spoofed. Attackers may easily forge the source IP address of the packets they send, but response packets will return to the forged IP address. To see the response packets, the attacker has to sniff the traffic between the victim machine and the forged IP address. In order to accomplish the required sniffing, attackers typically attempt to locate themselves on the same subnet as the victim machine. Attackers may be able to circumvent this requirement by using source routing, but source routing is disabled across much of the Internet today. In summary, IP address verification can be a useful part of an authentication scheme, but it should not be the single factor required for authentication.

Recommendation

You can increase confidence in a domain name lookup if you check to make sure that the host's forward and backward DNS entries match. Attackers will not be able to spoof both the forward and the reverse DNS entries without controlling the nameservers for the target domain. This is not a foolproof approach however: attackers may be able to convince the domain registrar to turn over the domain to a malicious nameserver. Basing authentication on DNS entries is simply a risky proposition. While no authentication mechanism is foolproof, there are better alternatives than host-based authentication. Password systems offer decent security, but are susceptible to bad password choices, insecure password transmission, and bad password management. A cryptographic scheme like SSL is worth considering, but such schemes are often so complex that they bring with them the risk of significant implementation errors, and key material can always be stolen. In many situations, multi-factor authentication including a physical token offers the most security available at a reasonable price.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Often Misused: Authentication	9	0	0	9
Total	**9**	**0**	**0**	**9**

Often Misused: Authentication — High

Package: akka.io

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 127 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getAllByName()
Enclosing Method: applyOrElse()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:127
Taint Flags:

```
124 case None =>
125 log.debug("Request for [{}] was not yet cached", name)
126 try {
127 val addresses: Array[InetAddress] = InetAddress.getAllByName(name)
128 val records = addressToRecords(name, addresses.toList, ipv4, ipv6)
129 val answer = DnsProtocol.Resolved(name, records.toList)
130 if (positiveCachePolicy != Never)
```

src/main/scala/akka/io/InetAddressDnsResolver.scala, line 148 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getAllByName()
Enclosing Method: applyOrElse()
File: src/main/scala/akka/io/InetAddressDnsResolver.scala:148
Taint Flags:

```
145 case Some(a) => a
146 case None =>
147 try {
148 val addresses = InetAddress.getAllByName(name)
149 // respond with the old protocol as the request was the new protocol
150 val answer = Dns.Resolved(name, addresses)
151 if (positiveCachePolicy != Never) {
```

Often Misused: Authentication — High

Package: akka.io.dns

src/main/scala/akka/io/dns/DnsResourceRecords.scala, line 42 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getByAddress()
Enclosing Method: parseBody()
File: src/main/scala/akka/io/dns/DnsResourceRecords.scala:42
Taint Flags:

```
39 def parseBody(name: String, ttl: Ttl, @unused length: Short, it: ByteIterator): ARecord = {
40 val address = Array.ofDim[Byte](4)
41 it.getBytes(address)
42 ARecord(name, ttl, InetAddress.getByAddress(address).asInstanceOf[Inet4Address])
43 }
44 }
45
```

src/main/scala/akka/io/dns/DnsResourceRecords.scala, line 62 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getByAddress()
Enclosing Method: parseBody()
File: src/main/scala/akka/io/dns/DnsResourceRecords.scala:62
Taint Flags:

```
59 def parseBody(name: String, ttl: Ttl, @unused length: Short, it: ByteIterator): AAAARecord = {
60 val address = Array.ofDim[Byte](16)
61 it.getBytes(address)
62 AAAARecord(name, ttl, InetAddress.getByAddress(address).asInstanceOf[Inet6Address])
63 }
64 }
65
```

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 46 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse

Often Misused: Authentication — High

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 46 (Often Misused: Authentication) — High

Scan Engine: SCA (Semantic)

Sink Details

Sink: getByName()
Enclosing Method: AsyncDnsResolver()
File: src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala:46
Taint Flags:

```
43 {
44 val loopback = InetAddress.getLoopbackAddress
45 val (ipv4Address, ipv6Address) = loopback match {
46 case ipv6: Inet6Address => (InetAddress.getByName("127.0.0.1"), ipv6)
47 case ipv4: Inet4Address => (ipv4, InetAddress.getByName("::1"))
48 case unknown => throw new IllegalArgumentException(s"Loopback address was [$unknown]")
49 }
```

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 47 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getByName()
Enclosing Method: AsyncDnsResolver()
File: src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala:47
Taint Flags:

```
44 val loopback = InetAddress.getLoopbackAddress
45 val (ipv4Address, ipv6Address) = loopback match {
46 case ipv6: Inet6Address => (InetAddress.getByName("127.0.0.1"), ipv6)
47 case ipv4: Inet4Address => (ipv4, InetAddress.getByName("::1"))
48 case unknown => throw new IllegalArgumentException(s"Loopback address was [$unknown]")
49 }
50 cache.put(
```

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 44 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Often Misused: Authentication — High

Package: akka.io.dns.internal

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 44 (Often Misused: Authentication) — High

Sink: getLoopbackAddress()
Enclosing Method: AsyncDnsResolver()
File: src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala:44
Taint Flags:

```
41
42 // avoid ever looking up localhost by pre-populating cache
43 {
44 val loopback = InetAddress.getLoopbackAddress
45 val (ipv4Address, ipv6Address) = loopback match {
46 case ipv6: Inet6Address => (InetAddress.getByName("127.0.0.1"), ipv6)
47 case ipv4: Inet4Address => (ipv4, InetAddress.getByName("::1"))
```

src/main/scala/akka/io/dns/internal/DnsClient.scala, line 53 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getByAddress()
Enclosing Method: preStart()
File: src/main/scala/akka/io/dns/internal/DnsClient.scala:53
Taint Flags:

```
50 lazy val tcpDnsClient: ActorRef = createTcpClient()
51
52 override def preStart() = {
53 udp ! Udp.Bind(self, new InetSocketAddress(InetAddress.getByAddress(Array.ofDim(4)), 0))
54 }
55
56 def receive: Receive = {
```

Package: src.main.scala.akka.io.dns.internal

src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 116 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getByName()
Enclosing Method: apply()

Often Misused: Authentication	High
Package: src.main.scala.akka.io.dns.internal	
src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala, line 116 (Often Misused: Authentication)	**High**

File: src/main/scala/akka/io/dns/internal/AsyncDnsResolver.scala:116
Taint Flags:

```
113 if (isInetAddress(name)) {
114 Future.fromTry {
115 Try {
116 val address = InetAddress.getByName(name) // only checks validity, since known to be IP address
117 val record = address match {
118 case _: Inet4Address => ARecord(name, Ttl.effectivelyForever, address)
119 case ipv6address: Inet6Address => AAAARecord(name, Ttl.effectivelyForever, ipv6address)
```

Poor Error Handling: Empty Catch Block (4 issues)

Abstract

Ignoring an exception can cause the program to overlook unexpected states and conditions.

Explanation

Just about every serious attack on a software system begins with the violation of a programmer's assumptions. After the attack, the programmer's assumptions seem flimsy and poorly founded, but before an attack many programmers would defend their assumptions well past the end of their lunch break. Two dubious assumptions that are easy to spot in code are "this method call can never fail" and "it doesn't matter if this call fails". When a programmer ignores an exception, they implicitly state that they are operating under one of these assumptions. **Example 1:** The following code excerpt ignores a rarely-thrown exception from `doExchange()`.

```
try {
  doExchange();
}
catch (RareException e) {
  // this can never happen
}
```

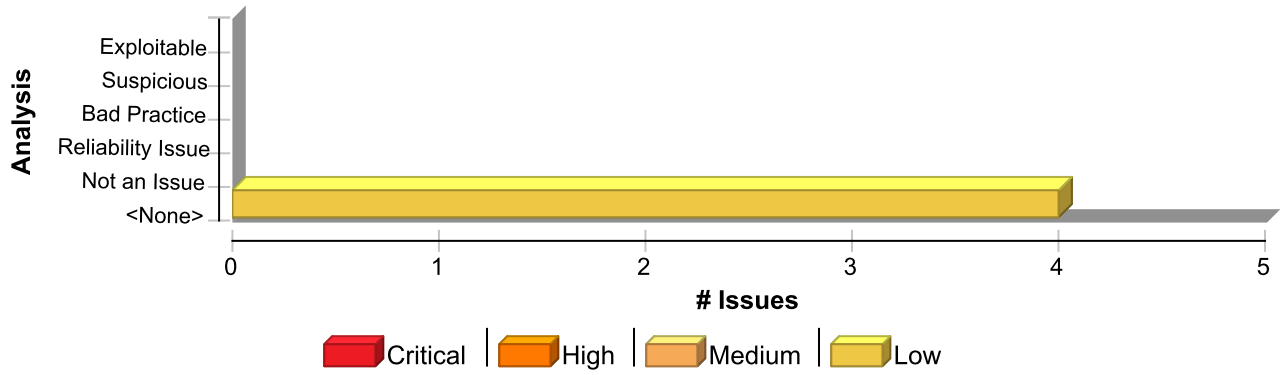
If a `RareException` were to ever be thrown, the program would continue to execute as though nothing unusual had occurred. The program records no evidence indicating the special situation, potentially frustrating any later attempt to explain the program's behavior.

Recommendation

At a minimum, log the fact that the exception was thrown so that it will be possible to come back later and make sense of the resulting program behavior. Better yet, abort the current operation. If the exception is being ignored because the caller cannot properly handle it but the context makes it inconvenient or impossible for the caller to declare that it throws the exception itself, consider throwing a `RuntimeException` or an `Error`, both of which are unchecked exceptions. As of JDK 1.4, `RuntimeException` has a constructor that makes it easy to wrap another exception. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
try {
  doExchange();
}
catch (RareException e) {
  throw new RuntimeException("This can never happen", e);
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Empty Catch Block	4	0	0	4
Total	**4**	**0**	**0**	**4**

Poor Error Handling: Empty Catch Block — Low

Package: akka.actor

src/main/scala/akka/actor/LightArrayRevolverScheduler.scala, line 118 (Poor Error Handling: Empty Catch Block) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: run()
File: src/main/scala/akka/actor/LightArrayRevolverScheduler.scala:118
Taint Flags:

```
115 if (self.get != null)
116 swap(schedule(executor, this, Duration.fromNanos(Math.max(delay.toNanos - driftNanos, 1))))
117 } catch {
118 case _: SchedulerException => // ignore failure to enqueue or terminated target actor
119 }
120 }
121 },
```

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 155 (Poor Error Handling: Empty Catch Block) — Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: startDefaultLoggers()
File: src/main/scala/akka/event/Logging.scala:155
Taint Flags:

```
152 }), "UnhandledMessageForwarder"),
153 classOf[UnhandledMessage])
154 } catch {
155 case _: InvalidActorNameException => // ignore if it is already running
156 }
157 publish(Debug(logName, this.getClass, "Default Loggers started"))
158 if (!(defaultLoggers contains StandardOutLogger.getClass.getName)) {
```

Poor Error Handling: Empty Catch Block	Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 155 (Poor Error Handling: Empty Catch Block)	Low

Package: akka.io

src/main/scala/akka/io/SelectionHandler.scala, line 172 (Poor Error Handling: Empty Catch Block)	Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: tryRun()
File: src/main/scala/akka/io/SelectionHandler.scala:172
Taint Flags:

```
169 case x => log.warning("Invalid readyOps: [{}]", x)
170 }
171 } catch {
172 case _: CancelledKeyException =>
173 // can be ignored because this exception is triggered when the key becomes invalid
174 // because `channel.close()` in `TcpConnection.postStop` is called from another thread
175 }
```

src/main/scala/akka/io/SelectionHandler.scala, line 207 (Poor Error Handling: Empty Catch Block)	Low

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: CatchBlock
Enclosing Method: tryRun()
File: src/main/scala/akka/io/SelectionHandler.scala:207
Taint Flags:

```
204 def cancelAndClose(andThen: () => Unit): Unit = cancelKeyAndClose(key, andThen)
205 }
206 } catch {
207 case _: ClosedChannelException =>
208 // ignore, might happen if a connection is closed in the same moment as an interest is registered
209 }
210 }
```

Poor Error Handling: Throw Inside Finally (1 issue)

Abstract

Using a `throw` statement inside a `finally` block breaks the logical progression through the `try-catch-finally`.

Explanation

In Java, `finally` blocks are always executed after their corresponding `try-catch` blocks and are often used to free allocated resources, such as file handles or database cursors. Throwing an exception in a `finally` block can bypass critical cleanup code since normal program execution will be disrupted. **Example 1:** In the following code, the call to `stmt.close()` is bypassed when the `FileNotFoundException` is thrown.

```
public void processTransaction(Connection conn) throws FileNotFoundException
{
    FileInputStream fis = null;
    Statement stmt = null;
    try
    {
        stmt = conn.createStatement();
        fis = new FileInputStream("badFile.txt");
        ...
    }
    catch (FileNotFoundException fe)
    {
        log("File not found.");
    }
    catch (SQLException se)
    {
        //handle error
    }
    finally
    {
        if (fis == null)
        {
            throw new FileNotFoundException();
        }

        if (stmt != null)
        {
            try
            {
                stmt.close();
            }
            catch (SQLException e)
            {
                log(e);
            }
        }
    }
}
```

This category is from the Cigital Java Rulepack.

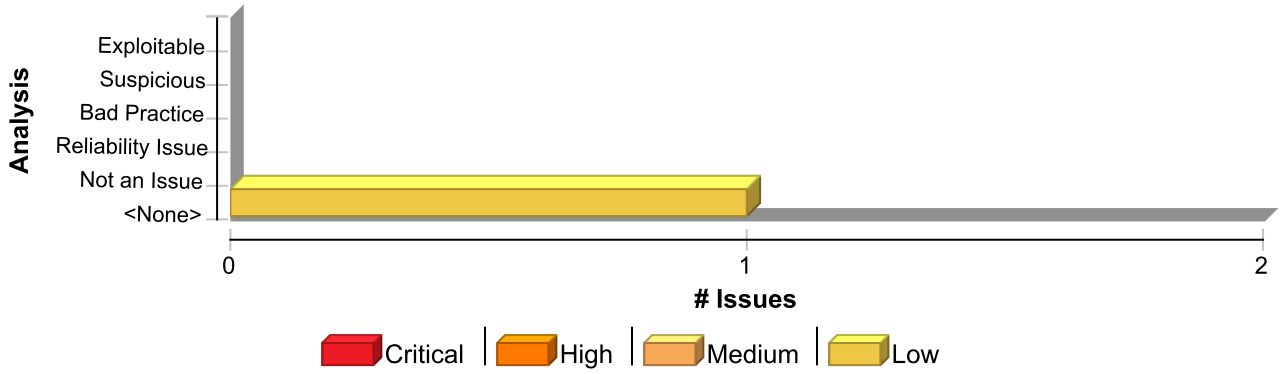
Recommendation

Never throw exceptions from within `finally` blocks. If you must re-throw an exception, do it inside a `catch` block so as not to interrupt the normal execution of the `finally` block. **Example 2:** The following code re-throws the

`FileNotFoundException` in the `catch` block.

```
public void processTransaction(Connection conn) throws FileNotFoundException
{
    FileInputStream fis = null;
    Statement stmt = null;
    try
    {
        stmt = conn.createStatement();
        fis = new FileInputStream("badFile.txt");
        ...
    }
    catch (FileNotFoundException fe)
    {
        log("File not found.");
        throw fe;
    }
    catch (SQLException se)
    {
        //handle error
    }
    finally
    {
        if (fis != null)
        {
            try
            {
                fis.close();
            }
            catch (IOException ie)
            {
                log(ie);
            }
        }

        if (stmt != null)
        {
            try
            {
                stmt.close();
            }
            catch (SQLException e)
            {
                log(e);
            }
        }
    }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Error Handling: Throw Inside Finally	1	0	0	1
Total	**1**	**0**	**0**	**1**

Poor Error Handling: Throw Inside Finally	**Low**
Package: akka.util	
src/main/scala/akka/util/LineNumbers.scala, line 160 (Poor Error Handling: Throw Inside Finally)	**Low**

Issue Details

Kingdom: Errors
Scan Engine: SCA (Structural)

Sink Details

Sink: FinallyBlock
Enclosing Method: getInfo()
File: src/main/scala/akka/util/LineNumbers.scala:160
Taint Flags:

```
157 private def getInfo(stream: InputStream, filter: Option[String]): Result = {
158 val dis = new DataInputStream(stream)
159
160 try {
161 skipID(dis)
162 skipVersion(dis)
163 implicit val constants = getConstants(dis)
```

Poor Logging Practice: Use of a System Output Stream (12 issues)

Abstract

Using `System.out` or `System.err` rather than a dedicated logging facility makes it difficult to monitor the program behavior.

Explanation

Example 1: The first Java program that a developer learns to write is the following:

```
public class MyClass
  ...
    System.out.println("hello world");
  ...
}
```

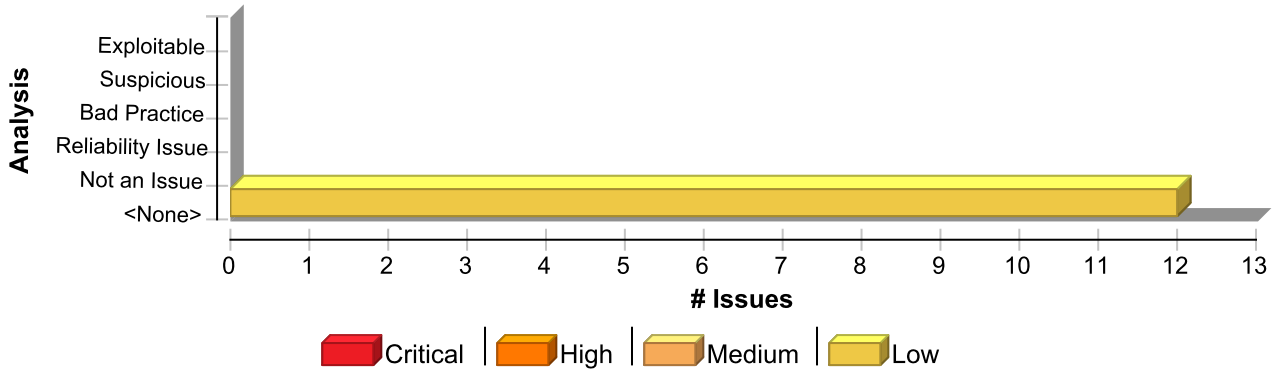
While most programmers go on to learn many nuances and subtleties about Java, a surprising number hang on to this first lesson and never give up on writing messages to standard output using `System.out.println()`. The problem is that writing directly to standard output or standard error is often used as an unstructured form of logging. Structured logging facilities provide features like logging levels, uniform formatting, a logger identifier, timestamps, and, perhaps most critically, the ability to direct the log messages to the right place. When the use of system output streams is jumbled together with the code that uses loggers properly, the result is often a well-kept log that is missing critical information. Developers widely accept the need for structured logging, but many continue to use system output streams in their "pre-production" development. If the code you are reviewing is past the initial phases of development, use of `System.out` or `System.err` may indicate an oversight in the move to a structured logging system.

Recommendation

Use a Java logging facility rather than `System.out` or `System.err`. **Example 2:** For example, you can rewrite the "hello world" program in `Example 1` using log4j as follows:

```
import org.apache.log4j.Logger;
import org.apache.log4j.BasicConfigurator;

public class MyClass {
  private final static Logger logger =
            Logger.getLogger(MyClass.class);

  ...
    BasicConfigurator.configure();
    logger.info("hello world");
  ...
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Logging Practice: Use of a System Output Stream	12	0	0	12
Total	**12**	**0**	**0**	**12**

Poor Logging Practice: Use of a System Output Stream — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 866 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: flush
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:866
Taint Flags:

```
863 err.print(" ActorSystem[")
864 err.print(name)
865 err.println("]")
866 System.err.flush()
867 cause.printStackTrace(System.err)
868 System.err.flush()
869
```

src/main/scala/akka/actor/ActorSystem.scala, line 868 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: flush

Poor Logging Practice: Use of a System Output Stream — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 868 (Poor Logging Practice: Use of a System Output Stream) — Low

Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:868
Taint Flags:

```
865 err.println("]")
866 System.err.flush()
867 cause.printStackTrace(System.err)
868 System.err.flush()
869
870 // Also log using the normal infrastructure - hope for the best:
871 markerLogging.error(
```

src/main/scala/akka/actor/ActorSystem.scala, line 864 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:864
Taint Flags:

```
861 err.print(", ")
862 err.print(message)
863 err.print(" ActorSystem[")
864 err.print(name)
865 err.println("]")
866 System.err.flush()
867 cause.printStackTrace(System.err)
```

src/main/scala/akka/actor/ActorSystem.scala, line 857 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:857
Taint Flags:

Poor Logging Practice: Use of a System Output Stream — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 857 (Poor Logging Practice: Use of a System Output Stream) — Low

```
854 private def logFatalError(message: String, cause: Throwable, thread: Thread): Unit = {
855 // First log to stderr as this has the best chance to get through in an 'emergency panic' situation:
856 import System.err
857 err.print("Uncaught error from thread [")
858 err.print(thread.getName)
859 err.print("]: ")
860 err.print(cause.getMessage)
```

src/main/scala/akka/actor/ActorSystem.scala, line 858 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:858
Taint Flags:

```
855 // First log to stderr as this has the best chance to get through in an 'emergency panic' situation:
856 import System.err
857 err.print("Uncaught error from thread [")
858 err.print(thread.getName)
859 err.print("]: ")
860 err.print(cause.getMessage)
861 err.print(", ")
```

src/main/scala/akka/actor/ActorSystem.scala, line 836 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: uncaughtException()
File: src/main/scala/akka/actor/ActorSystem.scala:836
Taint Flags:

```
833 log.error(cause, "Uncaught error from thread [{}]", thread.getName)
834 case _ =>
835 if (cause.isInstanceOf[IncompatibleClassChangeError] && cause.getMessage.startsWith("akka"))
```

Poor Logging Practice: Use of a System Output Stream — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 836 (Poor Logging Practice: Use of a System Output Stream) — Low

```
836 System.err.println(
837 s"""Detected ${cause.getClass.getName} error, which MAY be caused by incompatible Akka versions on the classpath.
838 | Please note that a given Akka version MUST be the same across all modules of Akka that you are using,
839 | e.g. if you use akka-actor [${akka.Version.current} (resolved from current classpath)] all other core
```

src/main/scala/akka/actor/ActorSystem.scala, line 860 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:860
Taint Flags:

```
857 err.print("Uncaught error from thread [")
858 err.print(thread.getName)
859 err.print("]: ")
860 err.print(cause.getMessage)
861 err.print(", ")
862 err.print(message)
863 err.print(" ActorSystem[")
```

src/main/scala/akka/actor/ActorSystem.scala, line 859 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:859
Taint Flags:

```
856 import System.err
857 err.print("Uncaught error from thread [")
858 err.print(thread.getName)
859 err.print("]: ")
860 err.print(cause.getMessage)
861 err.print(", ")
```

Poor Logging Practice: Use of a System Output Stream — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 859 (Poor Logging Practice: Use of a System Output Stream) — Low

```
862 err.print(message)
```

src/main/scala/akka/actor/ActorSystem.scala, line 861 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:861
Taint Flags:

```
858 err.print(thread.getName)
859 err.print("]: ")
860 err.print(cause.getMessage)
861 err.print(", ")
862 err.print(message)
863 err.print(" ActorSystem[")
864 err.print(name)
```

src/main/scala/akka/actor/ActorSystem.scala, line 865 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: println
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:865
Taint Flags:

```
862 err.print(message)
863 err.print(" ActorSystem[")
864 err.print(name)
865 err.println("]")
866 System.err.flush()
867 cause.printStackTrace(System.err)
868 System.err.flush()
```

Poor Logging Practice: Use of a System Output Stream — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 863 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:863
Taint Flags:

```
860 err.print(cause.getMessage)
861 err.print(", ")
862 err.print(message)
863 err.print(" ActorSystem[")
864 err.print(name)
865 err.println("]")
866 System.err.flush()
```

src/main/scala/akka/actor/ActorSystem.scala, line 862 (Poor Logging Practice: Use of a System Output Stream) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: print
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:862
Taint Flags:

```
859 err.print("]: ")
860 err.print(cause.getMessage)
861 err.print(", ")
862 err.print(message)
863 err.print(" ActorSystem[")
864 err.print(name)
865 err.println("]")
```

Poor Style: Confusing Naming (2 issues)

Abstract

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

Explanation

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

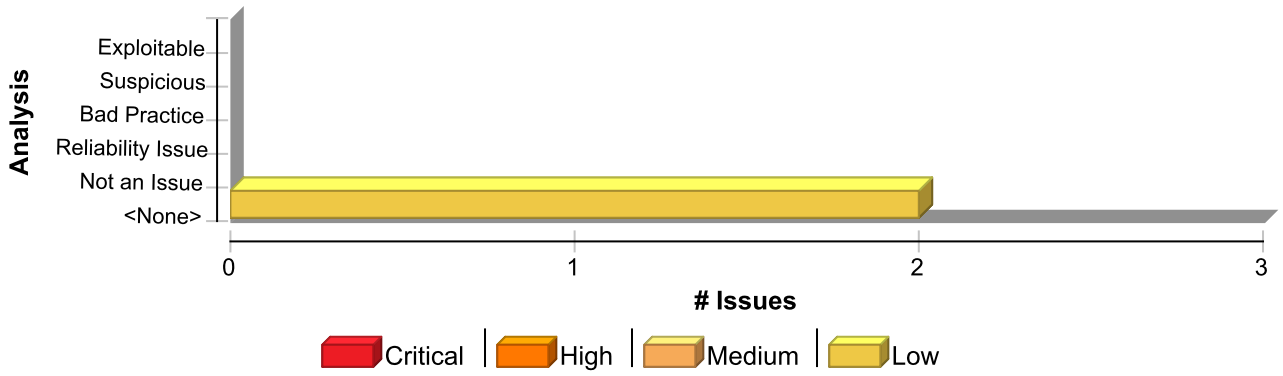
```
public class Totaller {
  private int total;
  public int total() {
    ...
  }
}
```

Recommendation

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

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

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Style: Confusing Naming	2	0	0	2
Total	**2**	**0**	**0**	**2**

Poor Style: Confusing Naming — Low

Package: akka.japi.pf

src/main/scala/akka/japi/pf/CaseStatements.scala, line 20 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: apply
File: src/main/scala/akka/japi/pf/CaseStatements.scala:20
Taint Flags:

```
17  override def apply(o: F) = apply.apply(o.asInstanceOf[P])
18 }
19
20 private[pf] class UnitCaseStatement[F, P](predicate: Predicate, apply: UnitApply[P]) extends PartialFunction[F, Unit] {
21
22  override def isDefinedAt(o: F) = predicate.defined(o)
23
```

src/main/scala/akka/japi/pf/CaseStatements.scala, line 13 (Poor Style: Confusing Naming) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Field: apply
File: src/main/scala/akka/japi/pf/CaseStatements.scala:13
Taint Flags:

```
10  def empty[F, T](): PartialFunction[F, T] = PartialFunction.empty
11 }
12
13 private[pf] class CaseStatement[-F, +P, T](predicate: Predicate, apply: Apply[P, T]) extends PartialFunction[F, T] {
14
15  override def isDefinedAt(o: F) = predicate.defined(o)
16
```

Poor Style: Value Never Read (10 issues)

Abstract

The variable's value is assigned but never used, making it a dead store.

Explanation

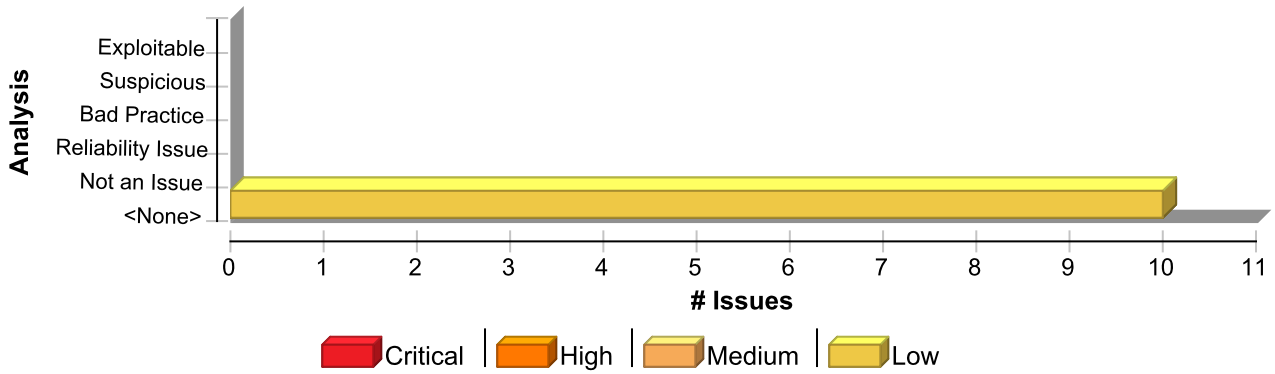
This variable's value is not used. After the assignment, the variable is either assigned another value or goes out of scope. **Example:** The following code excerpt assigns to the variable `r` and then overwrites the value without using it.

```
r = getName();
r = getNewBuffer(buf);
```

Recommendation

Remove unnecessary assignments in order to make the code easier to understand and maintain.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Poor Style: Value Never Read	10	0	0	10
Total	**10**	**0**	**0**	**10**

Poor Style: Value Never Read	**Low**
Package: akka.dispatch	
src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala, line 91 (Poor Style: Value Never Read)	**Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: tf
Enclosing Method: createExecutorServiceFactory()
File: src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala:91

Poor Style: Value Never Read — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/ForkJoinExecutorConfigurator.scala, line 91 (Poor Style: Value Never Read) — Low

Taint Flags:

```
88 case m: MonitorableThreadFactory =>
89 // add the dispatcher id to the thread names
90 m.withName(m.name + "-" + id)
91 case other => other
92 }
93
94 val asyncMode = config.getString("task-peeking-mode") match {
```

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 174 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: level
Enclosing Method: stopDefaultLoggers()
File: src/main/scala/akka/event/Logging.scala:174
Taint Flags:

```
171 */
172 private[akka] def stopDefaultLoggers(system: ActorSystem): Unit = {
173 @nowarn("msg=never used")
174 val level = _logLevel // volatile access before reading loggers
175 if (!(loggers contains StandardOutLogger)) {
176 setUpStdoutLogger(system.settings)
177 publish(Debug(simpleName(this), this.getClass, "shutting down: StandardOutLogger"))
```

Package: akka.util

src/main/scala/akka/util/LineNumbers.scala, line 225 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: major
Enclosing Method: skipVersion()
File: src/main/scala/akka/util/LineNumbers.scala:225
Taint Flags:

```
222
```

Poor Style: Value Never Read — Low

Package: akka.util

src/main/scala/akka/util/LineNumbers.scala, line 225 (Poor Style: Value Never Read) — Low

```
223 private def skipVersion(d: DataInputStream): Unit = {
224 val minor = d.readShort()
225 val major = d.readShort()
226 if (debug) println(s"LNB: version=$major:$minor")
227 }
228
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 298 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: stop
Enclosing Method: takeWhile()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:298
Taint Flags:

```
295 val lastLen = current.len
296 current.takeWhile(p)
297 if (current.hasNext) builder += current
298 if (current.len < lastLen) stop = true
299 dropCurrent()
300 }
301 iterators = builder.result()
```

src/main/scala-2.13/akka/util/ByteIterator.scala, line 97 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: stop
Enclosing Method: dropWhile()
File: src/main/scala-2.13/akka/util/ByteIterator.scala:97
Taint Flags:

```
94 if (p(array(from))) {
95 from = from + 1
96 } else {
97 stop = true
98 }
99 }
100 this
```

Poor Style: Value Never Read — Low

Package: akka.util

src/main/scala-2.13/akka/util/ByteIterator.scala, line 97 (Poor Style: Value Never Read) — Low

src/main/scala/akka/util/Version.scala, line 162 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: diff
Enclosing Method: compareTo()
File: src/main/scala/akka/util/Version.scala:162
Taint Flags:

```
159 diff = numbers(3) - other.numbers(3)
160 if (diff == 0) {
161 if (rest == "" && other.rest != "")
162 diff = 1
163 if (other.rest == "" && rest != "")
164 diff = -1
165 else
```

src/main/scala/akka/util/LineNumbers.scala, line 240 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: name
Enclosing Method: skipClassInfo()
File: src/main/scala/akka/util/LineNumbers.scala:240
Taint Flags:

```
237
238 private def skipClassInfo(d: DataInputStream)(implicit c: Constants): Unit = {
239 skip(d, 2) // access flags
240 val name = d.readUnsignedShort() // class name
241 skip(d, 2) // superclass name
242 if (debug) println(s"LNB: class name = ${c(name)}")
243 }
```

src/main/scala/akka/util/LineNumbers.scala, line 224 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Poor Style: Value Never Read — Low

Package: akka.util

src/main/scala/akka/util/LineNumbers.scala, line 224 (Poor Style: Value Never Read) — Low

Sink Details

Sink: VariableAccess: minor
Enclosing Method: skipVersion()
File: src/main/scala/akka/util/LineNumbers.scala:224
Taint Flags:

```
221 }
222
223 private def skipVersion(d: DataInputStream): Unit = {
224 val minor = d.readShort()
225 val major = d.readShort()
226 if (debug) println(s"LNB: version=$major:$minor")
227 }
```

src/main/scala/akka/util/LineNumbers.scala, line 261 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: name
Enclosing Method: skipMethodOrField()
File: src/main/scala/akka/util/LineNumbers.scala:261
Taint Flags:

```
258
259 private def skipMethodOrField(d: DataInputStream)(implicit c: Constants): Unit = {
260 skip(d, 2) // access flags
261 val name = d.readUnsignedShort() // name
262 skip(d, 2) // signature
263 val attributes = d.readUnsignedShort()
264 for (_ <- 1 to attributes) skipAttribute(d)
```

Package: src.main.scala.akka.util

src/main/scala/akka/util/LineNumbers.scala, line 248 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: intf
Enclosing Method: apply()
File: src/main/scala/akka/util/LineNumbers.scala:248
Taint Flags:

Poor Style: Value Never Read	Low

Package: src.main.scala.akka.util

src/main/scala/akka/util/LineNumbers.scala, line 248 (Poor Style: Value Never Read)	Low

```
245  private def skipInterfaceInfo(d: DataInputStream)(implicit c: Constants): Unit = {
246  val count = d.readUnsignedShort()
247  for (_ <- 1 to count) {
248  val intf = d.readUnsignedShort()
249  if (debug) println(s"LNB: implements ${c(intf)}")
250  }
251  }
```

Redundant Null Check (3 issues)

Abstract

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

Explanation

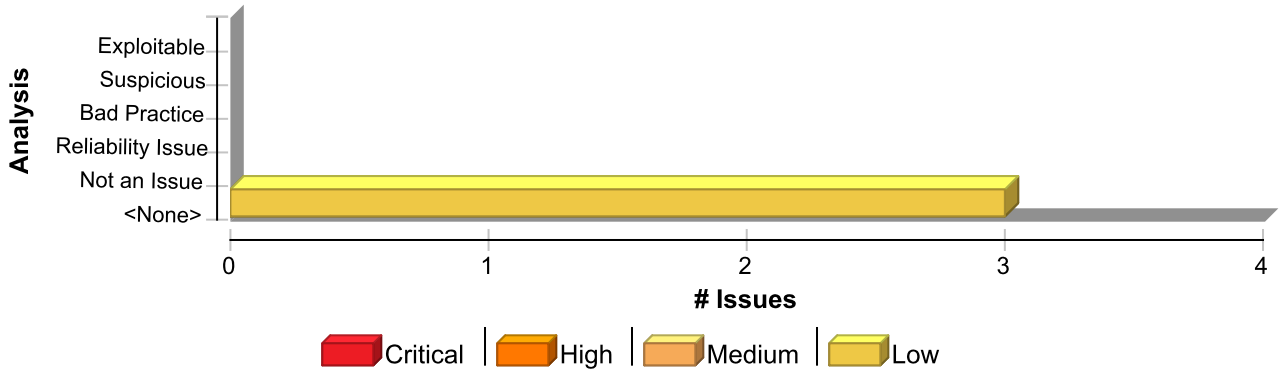
Null-pointer exceptions usually occur when one or more of the programmer's assumptions is violated. Specifically, dereference-after-check errors occur when a program makes an explicit check for `null`, but proceeds to dereference the object when it is known to be `null`. Errors of this type are often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

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

Recommendation

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

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Redundant Null Check	3	0	0	3
Total	**3**	**0**	**0**	**3**

Redundant Null Check — Low

Package: akka.actor.dungeon

src/main/scala/akka/actor/dungeon/FaultHandling.scala, line 129 (Redundant Null Check) — Low

Issue Details

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

Sink Details

Sink: Dereferenced : causedByFailure
Enclosing Method: faultResume()
File: src/main/scala/akka/actor/dungeon/FaultHandling.scala:129
Taint Flags:

```
126 Error(self.path.toString, clazz(actor), "changing Resume into Create after " + causedByFailure))
127 faultCreate()
128 } else if (isFailedFatally && causedByFailure != null) {
129 system.eventStream.publish(
130 Error(self.path.toString, clazz(actor), "changing Resume into Restart after " + causedByFailure))
131 faultRecreate(causedByFailure)
132 } else {
```

Package: akka.pattern

src/main/scala/akka/pattern/RetrySupport.scala, line 77 (Redundant Null Check) — Low

Issue Details

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

Sink Details

Sink: Dereferenced : minBackoff
Enclosing Method: retry()
File: src/main/scala/akka/pattern/RetrySupport.scala:77
Taint Flags:

```
74 require(attempt != null, "Parameter attempt should not be null.")
75 require(minBackoff != null, "Parameter minBackoff should not be null.")
76 require(maxBackoff != null, "Parameter maxBackoff should not be null.")
77 require(minBackoff > Duration.Zero, "Parameter minBackoff must be > 0")
78 require(maxBackoff >= minBackoff, "Parameter maxBackoff must be >= minBackoff")
79 require(0.0 <= randomFactor && randomFactor <= 1.0, "randomFactor must be between 0.0 and 1.0")
80 retry(
```

src/main/scala/akka/pattern/RetrySupport.scala, line 78 (Redundant Null Check) — Low

Issue Details

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

Sink Details

Redundant Null Check	Low
Package: akka.pattern	
src/main/scala/akka/pattern/RetrySupport.scala, line 78 (Redundant Null Check)	**Low**

Sink: Dereferenced : maxBackoff
Enclosing Method: retry()
File: src/main/scala/akka/pattern/RetrySupport.scala:78
Taint Flags:

```
75 require(minBackoff != null, "Parameter minBackoff should not be null.")
76 require(maxBackoff != null, "Parameter maxBackoff should not be null.")
77 require(minBackoff > Duration.Zero, "Parameter minBackoff must be > 0")
78 require(maxBackoff >= minBackoff, "Parameter maxBackoff must be >= minBackoff")
79 require(0.0 <= randomFactor && randomFactor <= 1.0, "randomFactor must be between 0.0 and 1.0")
80 retry(
81 attempt,
```

System Information Leak (5 issues)

Abstract

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

Explanation

An information leak occurs when system data or 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.

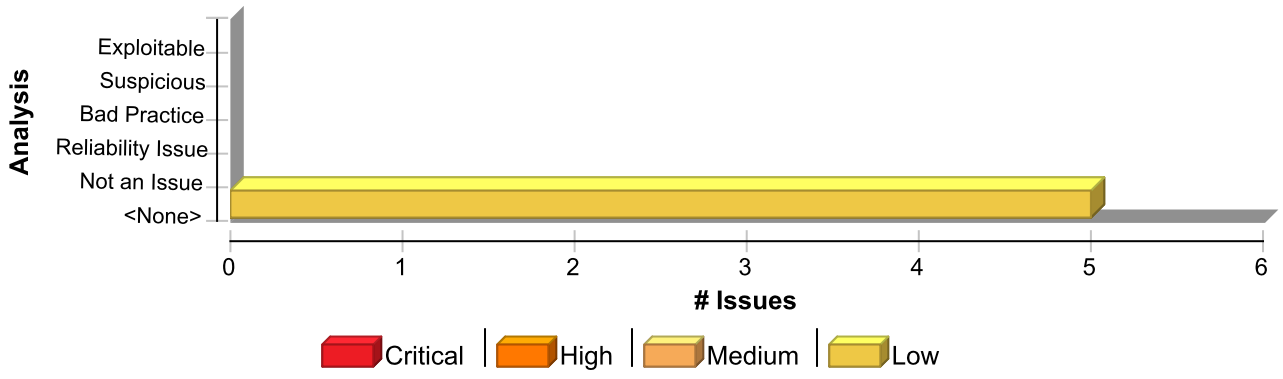
Recommendation

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	5	0	0	5
Total	**5**	**0**	**0**	**5**

System Information Leak — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 867 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:867
Taint Flags:

```
864 err.print(name)
865 err.println("]")
866 System.err.flush()
867 cause.printStackTrace(System.err)
868 System.err.flush()
869
870 // Also log using the normal infrastructure - hope for the best:
```

src/main/scala/akka/actor/ActorSelection.scala, line 365 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Runtime.availableProcessors()
From: akka.dispatch.ThreadPoolConfig.scaledPoolSize
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:38

```
35 val defaultRejectionPolicy: RejectedExecutionHandler = new SaneRejectedExecutionHandler()
36
37 def scaledPoolSize(floor: Int, multiplier: Double, ceiling: Int): Int =
38 math.min(math.max((Runtime.getRuntime.availableProcessors * multiplier).ceil.toInt, floor), ceiling)
39
40 def arrayBlockingQueue(capacity: Int, fair: Boolean): QueueFactory =
41 () => new ArrayBlockingQueue[Runnable](capacity, fair)
```

Sink Details

System Information Leak — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSelection.scala, line 365 (System Information Leak) — Low

Sink: java.lang.RuntimeException.RuntimeException()
Enclosing Method: ActorNotFound()
File: src/main/scala/akka/actor/ActorSelection.scala:365
Taint Flags: NUMBER, SYSTEMINFO

```
362 * `Future` is completed with this failure.
363 */
364 @SerialVersionUID(1L)
365 final case class ActorNotFound(selection: ActorSelection) extends RuntimeException("Actor not found for: " + selection)
366
367 undefined
368 undefined
```

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 1149 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: stackTraceFor()
File: src/main/scala/akka/event/Logging.scala:1149
Taint Flags:

```
1146 val sw = new java.io.StringWriter
1147 val pw = new java.io.PrintWriter(sw)
1148 pw.append('\n')
1149 other.printStackTrace(pw)
1150 sw.toString
1151 }
1152
```

src/main/scala/akka/event/Logging.scala, line 164 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: startDefaultLoggers()
File: src/main/scala/akka/event/Logging.scala:164
Taint Flags:

```
161 } catch {
```

System Information Leak — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 164 (System Information Leak) — Low

```
162 case e: Exception =>
163 System.err.println("error while starting up loggers")
164 e.printStackTrace()
165 throw new ConfigurationException("Could not start logger due to [" + e.toString + "]")
166 }
167 }
```

Package: src.main.scala.akka.actor

src/main/scala/akka/actor/ActorSelection.scala, line 74 (System Information Leak) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Runtime.availableProcessors()
From: akka.dispatch.ThreadPoolConfig.scaledPoolSize
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:38

```
35 val defaultRejectionPolicy: RejectedExecutionHandler = new SaneRejectedExecutionHandler()
36
37 def scaledPoolSize(floor: Int, multiplier: Double, ceiling: Int): Int =
38 math.min(math.max((Runtime.getRuntime.availableProcessors * multiplier).ceil.toInt, floor), ceiling)
39
40 def arrayBlockingQueue(capacity: Int, fair: Boolean): QueueFactory =
41 () => new ArrayBlockingQueue[Runnable](capacity, fair)
```

Sink Details

Sink: akka.actor.ActorNotFound.ActorNotFound()
Enclosing Method: apply()
File: src/main/scala/akka/actor/ActorSelection.scala:74
Taint Flags: NUMBER, SYSTEMINFO

```
71 val p = Promise[ActorRef]()
72 this.ask(Identify(None)).onComplete {
73 case Success(ActorIdentity(_, Some(ref))) => p.success(ref)
74 case _ => p.failure(ActorNotFound(this))
75 }
76 p.future
77 }
```

System Information Leak: External (33 issues)

Abstract

Revealing system data or debugging information could enable an adversary to use system information to plan an attack.

Explanation

An external information leak occurs when system data or debug information leaves the program to a remote machine via a socket or network connection. External leaks can help an attacker by revealing specific data about operating systems, full pathnames, the existence of usernames, or locations of configuration files, and are more serious than internal information leaks, which are more difficult for an attacker to access. **Example 1:** The following code leaks System details in the HTTP response:

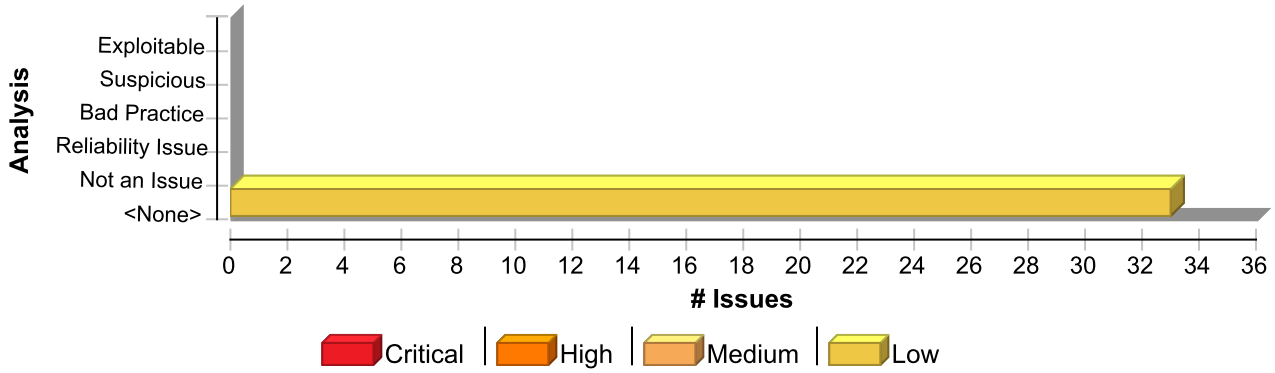
```
def doSomething() = Action { request =>
    ...
    Ok(Html(Properties.osName)) as HTML
}
```

This information can be exposed to a remote user. 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.

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.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak: External	33	0	0	33
Total	**33**	**0**	**0**	**33**

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 317 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:343

```
340  try {
341  val logMessage = cause match {
342  case e: ActorInitializationException if (e.getCause ne null) && (e.getCause.getMessage ne null) =>
343  e.getCause.getMessage
344  case e: ActorInitializationException if e.getCause ne null =>
345  e.getCause match {
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
```

Sink Details

Sink: akka.actor.Nobody.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:317
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
314 /**
315  * This is an internal look-up failure token, not useful for anything else.
316  */
317 private[akka] case object Nobody extends MinimalActorRef {
318  override val path: RootActorPath = new RootActorPath(Address("akka", "all-systems"), "/Nobody")
319  override def provider = throw new UnsupportedOperationException("Nobody does not provide")
320
```

src/main/scala/akka/actor/ActorRef.scala, line 793 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.dungeon.Dispatch$$anonfun$handleException$1.applyOrElse
File: src/main/scala/akka/actor/dungeon/Dispatch.scala:131

```
128  Thread.currentThread().interrupt()
129  case NonFatal(e) =>
```

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 793 (System Information Leak: External) — Low

```
130  val message = e match {
131  case n: NoStackTrace => "swallowing exception during message send: " + n.getMessage
132  case _ => "swallowing exception during message send" // stack trace includes message
133  }
134  system.eventStream.publish(Error(e, self.path.toString, clazz(actor), message))
```

Sink Details

Sink: akka.actor.VirtualPathContainer.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:793
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
790  emptyRef.tell(msg, sender)
791  }
792  }
793  case _ => super.!(message)
794  }
795
796  def addChild(name: String, ref: InternalActorRef): Unit = {
```

src/main/scala/akka/actor/ActorRef.scala, line 519 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.dungeon.Dispatch$$anonfun$handleException$1.applyOrElse
File: src/main/scala/akka/actor/dungeon/Dispatch.scala:131

```
128  Thread.currentThread().interrupt()
129  case NonFatal(e) =>
130  val message = e match {
131  case n: NoStackTrace => "swallowing exception during message send: " + n.getMessage
132  case _ => "swallowing exception during message send" // stack trace includes message
133  }
134  system.eventStream.publish(Error(e, self.path.toString, clazz(actor), message))
```

Sink Details

Sink: akka.actor.IgnoreActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:519
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 519 (System Information Leak: External) — Low

```
516 *
517 * INTERNAL API
518 */
519 @InternalApi private[akka] final class IgnoreActorRef(override val provider: ActorRefProvider) extends MinimalActorRef {
520
521 override val path: ActorPath = IgnoreActorRef.path
522
```

src/main/scala/akka/actor/ActorRef.scala, line 897 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Runtime.availableProcessors()
From: akka.dispatch.ThreadPoolConfig.scaledPoolSize
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:38

```
35 val defaultRejectionPolicy: RejectedExecutionHandler = new SaneRejectedExecutionHandler()
36
37 def scaledPoolSize(floor: Int, multiplier: Double, ceiling: Int): Int =
38 math.min(math.max((Runtime.getRuntime.availableProcessors * multiplier).ceil.toInt, floor), ceiling)
39
40 def arrayBlockingQueue(capacity: Int, fair: Boolean): QueueFactory =
41 () => new ArrayBlockingQueue[Runnable](capacity, fair)
```

Sink Details

Sink: akka.actor.FunctionRef.!()
Enclosing Method: sendSystemMessage()
File: src/main/scala/akka/actor/ActorRef.scala:897
Taint Flags: NUMBER, SYSTEMINFO

```
894 case w: Watch => addWatcher(w.watchee, w.watcher)
895 case u: Unwatch => remWatcher(u.watchee, u.watcher)
896 case DeathWatchNotification(actorRef, _, _) =>
897 this.!(Terminated(actorRef)(existenceConfirmed = true, addressTerminated = false))(actorRef)
898 case _ => //ignore all other messages
899 }
900 }
```

src/main/scala/akka/actor/ActorRef.scala, line 317 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 317 (System Information Leak: External) — Low

Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.dispatch.MessageDispatcher.reportFailure
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:176

```
173
174  override def reportFailure(t: Throwable): Unit = t match {
175  case e: LogEventException => eventStream.publish(e.event)
176  case _ => eventStream.publish(Error(t, getClass.getName, getClass, t.getMessage))
177  }
178
179  @tailrec
```

Sink Details

Sink: akka.actor.Nobody.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:317
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
314 /**
315  * This is an internal look-up failure token, not useful for anything else.
316  */
317 private[akka] case object Nobody extends MinimalActorRef {
318  override val path: RootActorPath = new RootActorPath(Address("akka", "all-systems"), "/Nobody")
319  override def provider = throw new UnsupportedOperationException("Nobody does not provide")
320
```

src/main/scala/akka/actor/ActorRef.scala, line 317 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:349

```
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
347  case t: Throwable => Logging.simpleName(t)
348  }
349  case e => e.getMessage
```

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 317 (System Information Leak: External) — Low

```
350  }
351  context.system.eventStream.publish(Logging.Debug(child.path.toString, classOf[SelectionHandler], logMessage))
352  } catch { case NonFatal(_) => }
```

Sink Details

Sink: akka.actor.Nobody.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:317
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
314 /**
315  * This is an internal look-up failure token, not useful for anything else.
316  */
317 private[akka] case object Nobody extends MinimalActorRef {
318  override val path: RootActorPath = new RootActorPath(Address("akka", "all-systems"), "/Nobody")
319  override def provider = throw new UnsupportedOperationException("Nobody does not provide")
320
```

src/main/scala/akka/actor/ActorRef.scala, line 519 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:343

```
340  try {
341  val logMessage = cause match {
342  case e: ActorInitializationException if (e.getCause ne null) && (e.getCause.getMessage ne null) =>
343  e.getCause.getMessage
344  case e: ActorInitializationException if e.getCause ne null =>
345  e.getCause match {
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
```

Sink Details

Sink: akka.actor.IgnoreActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:519
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
516 *
```

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 519 (System Information Leak: External) — Low

```
517 * INTERNAL API
518 */
519 @InternalApi private[akka] final class IgnoreActorRef(override val provider: ActorRefProvider) extends MinimalActorRef {
520
521 override val path: ActorPath = IgnoreActorRef.path
522
```

src/main/scala/akka/actor/ActorRef.scala, line 519 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:349

```
346 case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
347 case t: Throwable => Logging.simpleName(t)
348 }
349 case e => e.getMessage
350 }
351 context.system.eventStream.publish(Logging.Debug(child.path.toString, classOf[SelectionHandler], logMessage))
352 } catch { case NonFatal(_) => }
```

Sink Details

Sink: akka.actor.IgnoreActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:519
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
516 *
517 * INTERNAL API
518 */
519 @InternalApi private[akka] final class IgnoreActorRef(override val provider: ActorRefProvider) extends MinimalActorRef {
520
521 override val path: ActorPath = IgnoreActorRef.path
522
```

src/main/scala/akka/actor/ActorRef.scala, line 897 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 897 (System Information Leak: External) — Low

Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Runtime.availableProcessors()
From: akka.dispatch.ThreadPoolConfig.scaledPoolSize
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:38

```
35 val defaultRejectionPolicy: RejectedExecutionHandler = new SaneRejectedExecutionHandler()
36
37 def scaledPoolSize(floor: Int, multiplier: Double, ceiling: Int): Int =
38 math.min(math.max((Runtime.getRuntime.availableProcessors * multiplier).ceil.toInt, floor), ceiling)
39
40 def arrayBlockingQueue(capacity: Int, fair: Boolean): QueueFactory =
41 () => new ArrayBlockingQueue[Runnable](capacity, fair)
```

Sink Details

Sink: akka.actor.FunctionRef.!()
Enclosing Method: sendSystemMessage()
File: src/main/scala/akka/actor/ActorRef.scala:897
Taint Flags: NUMBER, SYSTEMINFO

```
894 case w: Watch => addWatcher(w.watchee, w.watcher)
895 case u: Unwatch => remWatcher(u.watchee, u.watcher)
896 case DeathWatchNotification(actorRef, _, _) =>
897 this.!(Terminated(actorRef)(existenceConfirmed = true, addressTerminated = false))(actorRef)
898 case _ => //ignore all other messages
899 }
900 }
```

src/main/scala/akka/actor/ActorRef.scala, line 519 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.dispatch.MessageDispatcher.reportFailure
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:176

```
173
174 override def reportFailure(t: Throwable): Unit = t match {
175 case e: LogEventException => eventStream.publish(e.event)
176 case _ => eventStream.publish(Error(t, getClass.getName, getClass, t.getMessage))
```

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 519 (System Information Leak: External) — Low

```
177  }
178
179  @tailrec
```

Sink Details

Sink: akka.actor.IgnoreActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:519
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
516 *
517 * INTERNAL API
518 */
519 @InternalApi private[akka] final class IgnoreActorRef(override val provider: ActorRefProvider) extends MinimalActorRef {
520
521  override val path: ActorPath = IgnoreActorRef.path
522
```

src/main/scala/akka/actor/ActorRef.scala, line 793 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.dispatch.MessageDispatcher.reportFailure
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:176

```
173
174  override def reportFailure(t: Throwable): Unit = t match {
175  case e: LogEventException => eventStream.publish(e.event)
176  case _ => eventStream.publish(Error(t, getClass.getName, getClass, t.getMessage))
177  }
178
179  @tailrec
```

Sink Details

Sink: akka.actor.VirtualPathContainer.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:793
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
790 emptyRef.tell(msg, sender)
791 }
```

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 793 (System Information Leak: External) — Low

```
792 }
793 case _ => super.!(message)
794 }
795
796 def addChild(name: String, ref: InternalActorRef): Unit = {
```

src/main/scala/akka/actor/ActorRef.scala, line 317 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.dungeon.Dispatch$$anonfun$handleException$1.applyOrElse
File: src/main/scala/akka/actor/dungeon/Dispatch.scala:131

```
128 Thread.currentThread().interrupt()
129 case NonFatal(e) =>
130 val message = e match {
131 case n: NoStackTrace => "swallowing exception during message send: " + n.getMessage
132 case _ => "swallowing exception during message send" // stack trace includes message
133 }
134 system.eventStream.publish(Error(e, self.path.toString, clazz(actor), message))
```

Sink Details

Sink: akka.actor.Nobody.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:317
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
314 /**
315 * This is an internal look-up failure token, not useful for anything else.
316 */
317 private[akka] case object Nobody extends MinimalActorRef {
318 override val path: RootActorPath = new RootActorPath(Address("akka", "all-systems"), "/Nobody")
319 override def provider = throw new UnsupportedOperationException("Nobody does not provide")
320
```

src/main/scala/akka/actor/ActorRef.scala, line 793 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 793 (System Information Leak: External) — Low

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:343

```
340  try {
341  val logMessage = cause match {
342  case e: ActorInitializationException if (e.getCause ne null) && (e.getCause.getMessage ne null) =>
343  e.getCause.getMessage
344  case e: ActorInitializationException if e.getCause ne null =>
345  e.getCause match {
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
```

Sink Details

Sink: akka.actor.VirtualPathContainer.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:793
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
790  emptyRef.tell(msg, sender)
791  }
792  }
793  case _ => super.!(message)
794  }
795
796  def addChild(name: String, ref: InternalActorRef): Unit = {
```

src/main/scala/akka/actor/ActorRef.scala, line 793 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:349

```
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
347  case t: Throwable => Logging.simpleName(t)
348  }
349  case e => e.getMessage
350  }
351  context.system.eventStream.publish(Logging.Debug(child.path.toString, classOf[SelectionHandler], logMessage))
```

System Information Leak: External — Low

Package: akka.actor

src/main/scala/akka/actor/ActorRef.scala, line 793 (System Information Leak: External) — Low

```
352  } catch { case NonFatal(_) => }
```

Sink Details

Sink: akka.actor.VirtualPathContainer.!()
Enclosing Method: !()
File: src/main/scala/akka/actor/ActorRef.scala:793
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
790  emptyRef.tell(msg, sender)
791  }
792  }
793  case _ => super.!(message)
794  }
795
796  def addChild(name: String, ref: InternalActorRef): Unit = {
```

Package: akka.event

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:349

```
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
347  case t: Throwable => Logging.simpleName(t)
348  }
349  case e => e.getMessage
350  }
351  context.system.eventStream.publish(Logging.Debug(child.path.toString, classOf[SelectionHandler],
logMessage))
352  } catch { case NonFatal(_) => }
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
```

System Information Leak: External — Low

Package: akka.event

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

```
41 protected def publish(event: Any, subscriber: ActorRef) = {
42 if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43 else subscriber ! event
44 }
45
46 override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.dungeon.FaultHandling$anonfun$faultRecreate$1.apply
File: src/main/scala/akka/actor/dungeon/FaultHandling.scala:95

```
92 if (!isFailedFatally) failedActor.aroundPreRestart(cause, optionalMessage)
93 } catch handleNonFatalOrInterruptedException { e =>
94 val ex = PreRestartException(self, e, cause, optionalMessage)
95 publish(Error(ex, self.path.toString, clazz(failedActor), e.getMessage))
96 } finally {
97 clearActorFields(failedActor, recreate = true)
98 }
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
41 protected def publish(event: Any, subscriber: ActorRef) = {
42 if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43 else subscriber ! event
44 }
45
46 override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

System Information Leak: External — Low

Package: akka.event

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.dungeon.FaultHandling$anonfun$finishTerminate$1.apply
File: src/main/scala/akka/actor/dungeon/FaultHandling.scala:242

```
239  */
240  try if (a ne null) a.aroundPostStop()
241  catch handleNonFatalOrInterruptedException { e =>
242  publish(Error(e, self.path.toString, clazz(a), e.getMessage))
243  } finally try stopFunctionRefs()
244  finally try dispatcher.detach(this)
245  finally try parent.sendSystemMessage(
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
41  protected def publish(event: Any, subscriber: ActorRef) = {
42  if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43  else subscriber ! event
44  }
45
46  override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.SupervisorStrategy.logFailure
File: src/main/scala/akka/actor/FaultHandling.scala:401

```
398  case ex: InvocationTargetException if ex.getCause ne null => ex.getCause.getMessage
399  case ex => ex.getMessage
400  }
401  case e => e.getMessage
402  }
403  decision match {
```

System Information Leak: External — Low

Package: akka.event

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

```
404  case Escalate => // don't log here
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
41  protected def publish(event: Any, subscriber: ActorRef) = {
42  if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43  else subscriber ! event
44  }
45
46  override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.event.Logging.stackTraceFor
File: src/main/scala/akka/event/Logging.scala:1144

```
1141  */
1142  def stackTraceFor(e: Throwable): String = e match {
1143  case null | Error.NoCause => ""
1144  case _: NoStackTrace => s" (${e.getClass.getName}: ${e.getMessage})"
1145  case other =>
1146  val sw = new java.io.StringWriter
1147  val pw = new java.io.PrintWriter(sw)
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
41  protected def publish(event: Any, subscriber: ActorRef) = {
42  if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43  else subscriber ! event
```

System Information Leak: External — Low

Package: akka.event

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

```
44 }
45
46 override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.dungeon.Dispatch$$anonfun$handleException$1.applyOrElse
File: src/main/scala/akka/actor/dungeon/Dispatch.scala:131

```
128 Thread.currentThread().interrupt()
129 case NonFatal(e) =>
130 val message = e match {
131 case n: NoStackTrace => "swallowing exception during message send: " + n.getMessage
132 case _ => "swallowing exception during message send" // stack trace includes message
133 }
134 system.eventStream.publish(Error(e, self.path.toString, clazz(actor), message))
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
41 protected def publish(event: Any, subscriber: ActorRef) = {
42 if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43 else subscriber ! event
44 }
45
46 override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

System Information Leak: External — Low

Package: akka.event

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Source: java.lang.Throwable.getMessage()
From: akka.dispatch.MessageDispatcher.reportFailure
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:176

```
173
174  override def reportFailure(t: Throwable): Unit = t match {
175  case e: LogEventException => eventStream.publish(e.event)
176  case _ => eventStream.publish(Error(t, getClass.getName, getClass, t.getMessage))
177  }
178
179  @tailrec
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
41  protected def publish(event: Any, subscriber: ActorRef) = {
42  if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43  else subscriber ! event
44  }
45
46  override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.SupervisorStrategy.logFailure
File: src/main/scala/akka/actor/FaultHandling.scala:399

```
396  case e: ActorInitializationException if e.getCause ne null =>
397  e.getCause match {
398  case ex: InvocationTargetException if ex.getCause ne null => ex.getCause.getMessage
399  case ex => ex.getMessage
400  }
401  case e => e.getMessage
402  }
```

System Information Leak: External — Low

Package: akka.event

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
41 protected def publish(event: Any, subscriber: ActorRef) = {
42 if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43 else subscriber ! event
44 }
45
46 override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.SupervisorStrategy.logFailure
File: src/main/scala/akka/actor/FaultHandling.scala:398

```
395 val logMessage = cause match {
396 case e: ActorInitializationException if e.getCause ne null =>
397 e.getCause match {
398 case ex: InvocationTargetException if ex.getCause ne null => ex.getCause.getMessage
399 case ex => ex.getMessage
400 }
401 case e => e.getMessage
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
41 protected def publish(event: Any, subscriber: ActorRef) = {
42 if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43 else subscriber ! event
44 }
45
```

System Information Leak: External — Low

Package: akka.event

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

```
46  override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

src/main/scala/akka/event/EventStream.scala, line 43 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:343

```
340  try {
341  val logMessage = cause match {
342  case e: ActorInitializationException if (e.getCause ne null) && (e.getCause.getMessage ne null) =>
343  e.getCause.getMessage
344  case e: ActorInitializationException if e.getCause ne null =>
345  e.getCause match {
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: publish()
File: src/main/scala/akka/event/EventStream.scala:43
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
40
41  protected def publish(event: Any, subscriber: ActorRef) = {
42  if (sys == null && subscriber.isTerminated) unsubscribe(subscriber)
43  else subscriber ! event
44  }
45
46  override def subscribe(subscriber: ActorRef, channel: Class[_]): Boolean = {
```

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 613 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

System Information Leak: External — Low

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 613 (System Information Leak: External) — Low

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:349

```
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
347  case t: Throwable => Logging.simpleName(t)
348  }
349  case e => e.getMessage
350  }
351  context.system.eventStream.publish(Logging.Debug(child.path.toString, classOf[SelectionHandler],
logMessage))
352  } catch { case NonFatal(_) => }
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/pattern/AskSupport.scala:613
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
610
611  override def !(message: Any)(implicit sender: ActorRef = Actor.noSender): Unit = state match {
612  case Stopped | _: StoppedWithPath =>
613  provider.deadLetters ! message
614  onComplete(message, alreadyCompleted = true)
615  case _ =>
616  if (message == null) throw InvalidMessageException("Message is null")
```

src/main/scala/akka/pattern/AskSupport.scala, line 624 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:349

```
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
347  case t: Throwable => Logging.simpleName(t)
348  }
349  case e => e.getMessage
350  }
```

System Information Leak: External — Low

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 624 (System Information Leak: External) — Low

```
351  context.system.eventStream.publish(Logging.Debug(child.path.toString, classOf[SelectionHandler],
logMessage))
352  } catch { case NonFatal(_) => }
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/pattern/AskSupport.scala:624
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
621  }
622  val alreadyCompleted = !result.tryComplete(promiseResult)
623  if (alreadyCompleted)
624  provider.deadLetters ! message
625  onComplete(message, alreadyCompleted)
626  }
627
```

src/main/scala/akka/pattern/AskSupport.scala, line 613 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:343

```
340  try {
341  val logMessage = cause match {
342  case e: ActorInitializationException if (e.getCause ne null) && (e.getCause.getMessage ne null) =>
343  e.getCause.getMessage
344  case e: ActorInitializationException if e.getCause ne null =>
345  e.getCause match {
346  case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/pattern/AskSupport.scala:613
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

System Information Leak: External — Low

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 613 (System Information Leak: External) — Low

```
610
611 override def !(message: Any)(implicit sender: ActorRef = Actor.noSender): Unit = state match {
612 case Stopped | _: StoppedWithPath =>
613 provider.deadLetters ! message
614 onComplete(message, alreadyCompleted = true)
615 case _ =>
616 if (message == null) throw InvalidMessageException("Message is null")
```

src/main/scala/akka/pattern/AskSupport.scala, line 624 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.io.SelectionHandler$$anon$10.logFailure
File: src/main/scala/akka/io/SelectionHandler.scala:343

```
340 try {
341 val logMessage = cause match {
342 case e: ActorInitializationException if (e.getCause ne null) && (e.getCause.getMessage ne null) =>
343 e.getCause.getMessage
344 case e: ActorInitializationException if e.getCause ne null =>
345 e.getCause match {
346 case ie: java.lang.reflect.InvocationTargetException => ie.getTargetException.toString
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/pattern/AskSupport.scala:624
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
621 }
622 val alreadyCompleted = !result.tryComplete(promiseResult)
623 if (alreadyCompleted)
624 provider.deadLetters ! message
625 onComplete(message, alreadyCompleted)
626 }
627
```

System Information Leak: External — Low

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 613 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.dispatch.MessageDispatcher.reportFailure
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:176

```
173
174  override def reportFailure(t: Throwable): Unit = t match {
175  case e: LogEventException => eventStream.publish(e.event)
176  case _ => eventStream.publish(Error(t, getClass.getName, getClass, t.getMessage))
177  }
178
179  @tailrec
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/pattern/AskSupport.scala:613
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
610
611  override def !(message: Any)(implicit sender: ActorRef = Actor.noSender): Unit = state match {
612  case Stopped | _: StoppedWithPath =>
613  provider.deadLetters ! message
614  onComplete(message, alreadyCompleted = true)
615  case _ =>
616  if (message == null) throw InvalidMessageException("Message is null")
```

src/main/scala/akka/pattern/AskSupport.scala, line 624 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.dispatch.MessageDispatcher.reportFailure
File: src/main/scala/akka/dispatch/AbstractDispatcher.scala:176

```
173
```

System Information Leak: External — Low

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 624 (System Information Leak: External) — Low

```
174 override def reportFailure(t: Throwable): Unit = t match {
175 case e: LogEventException => eventStream.publish(e.event)
176 case _ => eventStream.publish(Error(t, getClass.getName, getClass, t.getMessage))
177 }
178
179 @tailrec
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/pattern/AskSupport.scala:624
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
621 }
622 val alreadyCompleted = !result.tryComplete(promiseResult)
623 if (alreadyCompleted)
624 provider.deadLetters ! message
625 onComplete(message, alreadyCompleted)
626 }
627
```

src/main/scala/akka/pattern/AskSupport.scala, line 630 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Runtime.availableProcessors()
From: akka.dispatch.ThreadPoolConfig.scaledPoolSize
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:38

```
35 val defaultRejectionPolicy: RejectedExecutionHandler = new SaneRejectedExecutionHandler()
36
37 def scaledPoolSize(floor: Int, multiplier: Double, ceiling: Int): Int =
38 math.min(math.max((Runtime.getRuntime.availableProcessors * multiplier).ceil.toInt, floor), ceiling)
39
40 def arrayBlockingQueue(capacity: Int, fair: Boolean): QueueFactory =
41 () => new ArrayBlockingQueue[Runnable](capacity, fair)
```

Sink Details

System Information Leak: External — Low

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 630 (System Information Leak: External) — Low

Sink: akka.pattern.PromiseActorRef.!()
Enclosing Method: sendSystemMessage()
File: src/main/scala/akka/pattern/AskSupport.scala:630
Taint Flags: NUMBER, SYSTEMINFO

```
627
628 override def sendSystemMessage(message: SystemMessage): Unit = message match {
629 case _: Terminate => stop()
630 case DeathWatchNotification(a, ec, at) => this.!(Terminated(a)(existenceConfirmed = ec, addressTerminated = at))
631 case Watch(watchee, watcher) =>
632 if (watchee == this && watcher != this) {
633 if (!addWatcher(watcher))
```

src/main/scala/akka/pattern/AskSupport.scala, line 613 (System Information Leak: External) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.dungeon.Dispatch$$anonfun$handleException$1.applyOrElse
File: src/main/scala/akka/actor/dungeon/Dispatch.scala:131

```
128 Thread.currentThread().interrupt()
129 case NonFatal(e) =>
130 val message = e match {
131 case n: NoStackTrace => "swallowing exception during message send: " + n.getMessage
132 case _ => "swallowing exception during message send" // stack trace includes message
133 }
134 system.eventStream.publish(Error(e, self.path.toString, clazz(actor), message))
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/pattern/AskSupport.scala:613
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
610
611 override def !(message: Any)(implicit sender: ActorRef = Actor.noSender): Unit = state match {
612 case Stopped | _: StoppedWithPath =>
613 provider.deadLetters ! message
614 onComplete(message, alreadyCompleted = true)
615 case _ =>
```

System Information Leak: External	Low

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 613 (System Information Leak: External) **Low**

```
616 if (message == null) throw InvalidMessageException("Message is null")
```

src/main/scala/akka/pattern/AskSupport.scala, line 624 (System Information Leak: External) **Low**

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.dungeon.Dispatch$$anonfun$handleException$1.applyOrElse
File: src/main/scala/akka/actor/dungeon/Dispatch.scala:131

```
128 Thread.currentThread().interrupt()
129 case NonFatal(e) =>
130 val message = e match {
131 case n: NoStackTrace => "swallowing exception during message send: " + n.getMessage
132 case _ => "swallowing exception during message send" // stack trace includes message
133 }
134 system.eventStream.publish(Error(e, self.path.toString, clazz(actor), message))
```

Sink Details

Sink: akka.actor.DeadLetterActorRef.!()
Enclosing Method: !()
File: src/main/scala/akka/pattern/AskSupport.scala:624
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
621 }
622 val alreadyCompleted = !result.tryComplete(promiseResult)
623 if (alreadyCompleted)
624 provider.deadLetters ! message
625 onComplete(message, alreadyCompleted)
626 }
627
```

System Information Leak: Internal (5 issues)

Abstract

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

Explanation

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

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

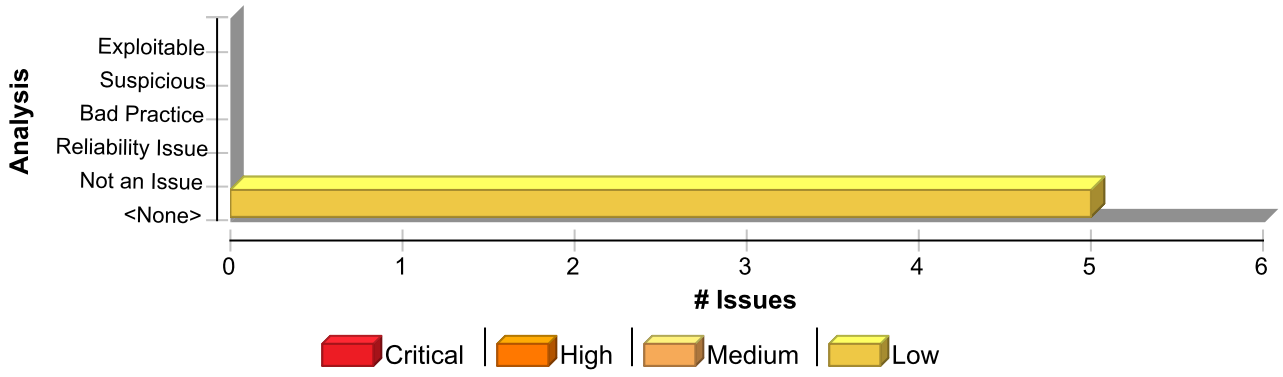
Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a user. In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal 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. **Example 2:** The following code logs the stack trace of a caught exception on the Android platform.

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

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. 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.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak: Internal	5	0	0	5
Total	**5**	**0**	**0**	**5**

System Information Leak: Internal — Low

Package: akka.actor

src/main/scala/akka/actor/ActorSystem.scala, line 860 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.actor.ActorSystemImpl$$anon$1.logFatalError
File: src/main/scala/akka/actor/ActorSystem.scala:860

```
857  err.print("Uncaught error from thread [")
858  err.print(thread.getName)
859  err.print("]: ")
860  err.print(cause.getMessage)
861  err.print(", ")
862  err.print(message)
863  err.print(" ActorSystem[")
```

Sink Details

Sink: java.io.PrintStream.print()
Enclosing Method: logFatalError()
File: src/main/scala/akka/actor/ActorSystem.scala:860
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
857  err.print("Uncaught error from thread [")
858  err.print(thread.getName)
859  err.print("]: ")
860  err.print(cause.getMessage)
861  err.print(", ")
862  err.print(message)
863  err.print(" ActorSystem[")
```

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 1011 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

System Information Leak: Internal — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 1011 (System Information Leak: Internal) — Low

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.event.Logging.stackTraceFor
File: src/main/scala/akka/event/Logging.scala:1144

```
1141  */
1142  def stackTraceFor(e: Throwable): String = e match {
1143  case null | Error.NoCause => ""
1144  case _: NoStackTrace => s" (${e.getClass.getName}: ${e.getMessage})"
1145  case other =>
1146  val sw = new java.io.StringWriter
1147  val pw = new java.io.PrintWriter(sw)
```

Sink Details

Sink: scala.Predef.println()
Enclosing Method: error()
File: src/main/scala/akka/event/Logging.scala:1011
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
1008  stackTraceFor(event.cause)))
1009  case _ =>
1010  val f = if (event.cause == Error.NoCause) ErrorFormatWithoutCause else ErrorFormat
1011  println(
1012  f.format(
1013  timestamp(event),
1014  event.thread.getName,
```

src/main/scala/akka/event/Logging.scala, line 1000 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.event.Logging.stackTraceFor
File: src/main/scala/akka/event/Logging.scala:1144

```
1141  */
1142  def stackTraceFor(e: Throwable): String = e match {
1143  case null | Error.NoCause => ""
1144  case _: NoStackTrace => s" (${e.getClass.getName}: ${e.getMessage})"
1145  case other =>
1146  val sw = new java.io.StringWriter
```

System Information Leak: Internal — Low

Package: akka.event

src/main/scala/akka/event/Logging.scala, line 1000 (System Information Leak: Internal) — Low

```
1147  val pw = new java.io.PrintWriter(sw)
```

Sink Details

Sink: scala.Predef.println()
Enclosing Method: error()
File: src/main/scala/akka/event/Logging.scala:1000
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
997  def error(event: Error): Unit = event match {
998  case e: Error3 => // has marker
999  val f = if (event.cause == Error.NoCause) ErrorWithoutCauseWithMarkerFormat else ErrorFormatWithMarker
1000  println(
1001  f.format(
1002  e.marker.name,
1003  timestamp(event),
```

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 637 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Runtime.availableProcessors()
From: akka.dispatch.ThreadPoolConfig.scaledPoolSize
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:38

```
35  val defaultRejectionPolicy: RejectedExecutionHandler = new SaneRejectedExecutionHandler()
36
37  def scaledPoolSize(floor: Int, multiplier: Double, ceiling: Int): Int =
38  math.min(math.max((Runtime.getRuntime.availableProcessors * multiplier).ceil.toInt, floor), ceiling)
39
40  def arrayBlockingQueue(capacity: Int, fair: Boolean): QueueFactory =
41  () => new ArrayBlockingQueue[Runnable](capacity, fair)
```

Sink Details

Sink: java.io.PrintStream.println()
Enclosing Method: sendSystemMessage()
File: src/main/scala/akka/pattern/AskSupport.scala:637
Taint Flags: NUMBER, SYSTEMINFO

```
634  // ➡➡➡ NEVER SEND THE SAME SYSTEM MESSAGE OBJECT TO TWO ACTORS
```

System Information Leak: Internal — Low

Package: akka.pattern

src/main/scala/akka/pattern/AskSupport.scala, line 637 (System Information Leak: Internal) — Low

```
635 watcher.sendSystemMessage(
636 DeathWatchNotification(watchee, existenceConfirmed = true, addressTerminated = false))
637 } else System.err.println("BUG: illegal Watch(%s,%s) for %s".format(watchee, watcher, this))
638 case Unwatch(watchee, watcher) =>
639 if (watchee == this && watcher != this) remWatcher(watcher)
640 else System.err.println("BUG: illegal Unwatch(%s,%s) for %s".format(watchee, watcher, this))
```

src/main/scala/akka/pattern/AskSupport.scala, line 640 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Runtime.availableProcessors()
From: akka.dispatch.ThreadPoolConfig.scaledPoolSize
File: src/main/scala/akka/dispatch/ThreadPoolBuilder.scala:38

```
35 val defaultRejectionPolicy: RejectedExecutionHandler = new SaneRejectedExecutionHandler()
36
37 def scaledPoolSize(floor: Int, multiplier: Double, ceiling: Int): Int =
38 math.min(math.max((Runtime.getRuntime.availableProcessors * multiplier).ceil.toInt, floor), ceiling)
39
40 def arrayBlockingQueue(capacity: Int, fair: Boolean): QueueFactory =
41 () => new ArrayBlockingQueue[Runnable](capacity, fair)
```

Sink Details

Sink: java.io.PrintStream.println()
Enclosing Method: sendSystemMessage()
File: src/main/scala/akka/pattern/AskSupport.scala:640
Taint Flags: NUMBER, SYSTEMINFO

```
637 } else System.err.println("BUG: illegal Watch(%s,%s) for %s".format(watchee, watcher, this))
638 case Unwatch(watchee, watcher) =>
639 if (watchee == this && watcher != this) remWatcher(watcher)
640 else System.err.println("BUG: illegal Unwatch(%s,%s) for %s".format(watchee, watcher, this))
641 case _ =>
642 }
643
```

Unchecked Return Value (1 issue)

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.

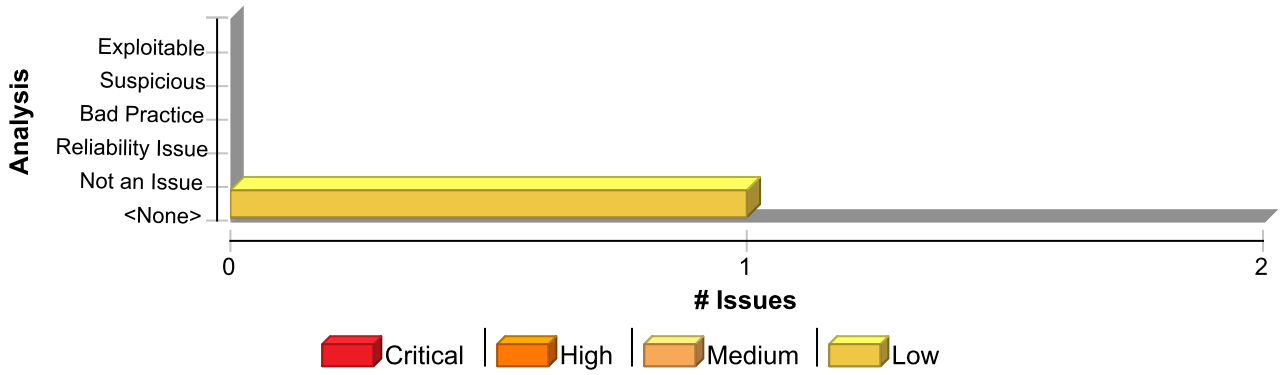
```
FileInputStream fis;
byte[] byteArray = new byte[1024];
for (Iterator i=users.iterator(); i.hasNext();) {
    String userName = (String) i.next();
    String pFileName = PFILE_ROOT + "/" + userName;
    FileInputStream fis = new FileInputStream(pFileName);
    fis.read(byteArray); // the file is 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	1	0	0	1
Total	**1**	**0**	**0**	**1**

Unchecked Return Value — Low

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 327 (Unchecked Return Value) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: interrupted()
Enclosing Method: processAllSystemMessages()
File: src/main/scala/akka/dispatch/Mailbox.scala:327
Taint Flags:

```
324 }
325 // if we got an interrupted exception while handling system messages, then rethrow it
326 if (interruption ne null) {
327 Thread.interrupted() // clear interrupted flag before throwing according to java convention
328 throw interruption
329 }
330 }
```

Unreleased Resource: Streams (2 issues)

Abstract

The program can potentially fail to release a system resource.

Explanation

The program can potentially fail to release a system resource. Resource leaks have at least two common causes: - Error conditions and other exceptional circumstances. - Confusion over which part of the program is responsible for releasing the resource. Most unreleased resource issues result in general software reliability problems. However, if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service attack by depleting the resource pool. **Example:** The following method never closes the file handle it opens. The `finalize()` method for `FileInputStream` eventually calls `close()`, but there is no guarantee as to how long it will take before the `finalize()` method will be invoked. In a busy environment, this can result in the JVM using up all of its file handles.

```
private void processFile(String fName) throws FileNotFoundException,
IOException {
  FileInputStream fis = new FileInputStream(fName);
  int sz;
  byte[] byteArray = new byte[BLOCK_SIZE];
  while ((sz = fis.read(byteArray)) != -1) {
    processBytes(byteArray, sz);
  }
}
```

Recommendation

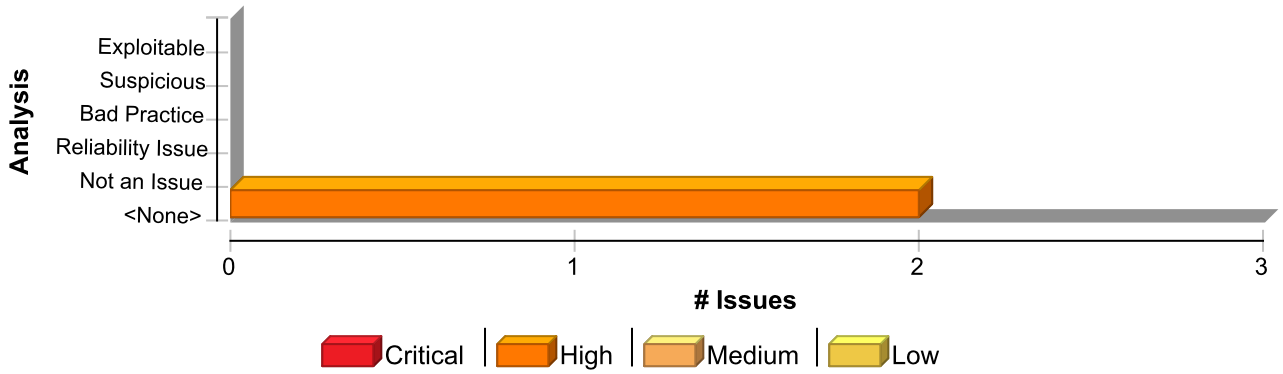
1. Never rely on `finalize()` to reclaim resources. In order for an object's `finalize()` method to be invoked, the garbage collector must determine that the object is eligible for garbage collection. Because the garbage collector is not required to run unless the JVM is low on memory, there is no guarantee that an object's `finalize()` method will be invoked in an expedient fashion. When the garbage collector finally does run, it may cause a large number of resources to be reclaimed in a short period of time, which can lead to "bursty" performance and lower overall system throughput. This effect becomes more pronounced as the load on the system increases. Finally, if it is possible for a resource reclamation operation to hang (if it requires communicating over a network to a database, for example), then the thread that is executing the `finalize()` method will hang. 2. Release resources in a `finally` block. The code for the Example should be rewritten as follows:

```
public void processFile(String fName) throws FileNotFoundException,
IOException {
  FileInputStream fis;
  try {
    fis = new FileInputStream(fName);
    int sz;
    byte[] byteArray = new byte[BLOCK_SIZE];
    while ((sz = fis.read(byteArray)) != -1) {
      processBytes(byteArray, sz);
    }
  }
  finally {
    if (fis != null) {
      safeClose(fis);
    }
  }
}
```

```
public static void safeClose(FileInputStream fis) {
  if (fis != null) {
    try {
      fis.close();
    } catch (IOException e) {
      log(e);
    }
  }
}
```

This solution uses a helper function to log the exceptions that might occur when trying to close the stream. Presumably this helper function will be reused whenever a stream needs to be closed. Also, the `processFile` method does not initialize the `fis` object to `null`. Instead, it checks to ensure that `fis` is not `null` before calling `safeClose()`. Without the `null` check, the Java compiler reports that `fis` might not be initialized. This choice takes advantage of Java's ability to detect uninitialized variables. If `fis` is initialized to `null` in a more complex method, cases in which `fis` is used without being initialized will not be detected by the compiler.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Streams	2	0	0	2
Total	**2**	**0**	**0**	**2**

Unreleased Resource: Streams	**High**
Package: akka.util	
src/main/scala/akka/util/LineNumbers.scala, line 193 (Unreleased Resource: Streams)	**High**

Issue Details

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

Sink Details

Sink: r = getResourceAsStream(...)
Enclosing Method: getStreamForClass()
File: src/main/scala/akka/util/LineNumbers.scala:193
Taint Flags:

```
190 private def getStreamForClass(c: Class[_]): Option[(InputStream, None.type)] = {
191 val resource = c.getName.replace('.', '/') + ".class"
```

Unreleased Resource: Streams — High

Package: akka.util

src/main/scala/akka/util/LineNumbers.scala, line 193 (Unreleased Resource: Streams) — High

```
192 val cl = c.getClassLoader
193 val r = cl.getResourceAsStream(resource)
194 if (debug) println(s"LNB: resource '$resource' resolved to stream $r")
195 Option(r).map(_ -> None)
196 }
```

src/main/scala/akka/util/LineNumbers.scala, line 204 (Unreleased Resource: Streams) — High

Issue Details

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

Sink Details

Sink: getResourceAsStream(...)
Enclosing Method: getStreamForLambda()
File: src/main/scala/akka/util/LineNumbers.scala:204
Taint Flags:

```
201 val writeReplace = c.getDeclaredMethod("writeReplace")
202 writeReplace.setAccessible(true)
203 writeReplace.invoke(l) match {
204 case serialized: SerializedLambda =>
205 if (debug)
206 println(s"LNB: found Lambda implemented in ${serialized.getImplClass}:${serialized.getImplMethodName}")
207 Option(c.getClassLoader.getResourceAsStream(serialized.getImplClass + ".class"))
```

Unreleased Resource: Synchronization (23 issues)

Abstract

The program fails to release a lock it holds, which might lead to deadlock.

Explanation

The program can potentially fail to release a system resource. Resource leaks have at least two common causes: - Error conditions and other exceptional circumstances. - Confusion over which part of the program is responsible for releasing the resource. Most unreleased resource issues result in general software reliability problems. However, if an attacker can intentionally trigger a resource leak, the attacker may be able to launch a denial of service by depleting the resource pool. **Example 1:** The following code establishes a lock before `performOperationInCriticalSection()`, but fails to release the lock if an exception is thrown in that method.

```
  ReentrantLock  myLock = new ReentrantLock();

  myLock.lock();
  performOperationInCriticalSection();
  myLock.unlock();
```

This category was derived from the Cigital Java Rulepack.

Recommendation

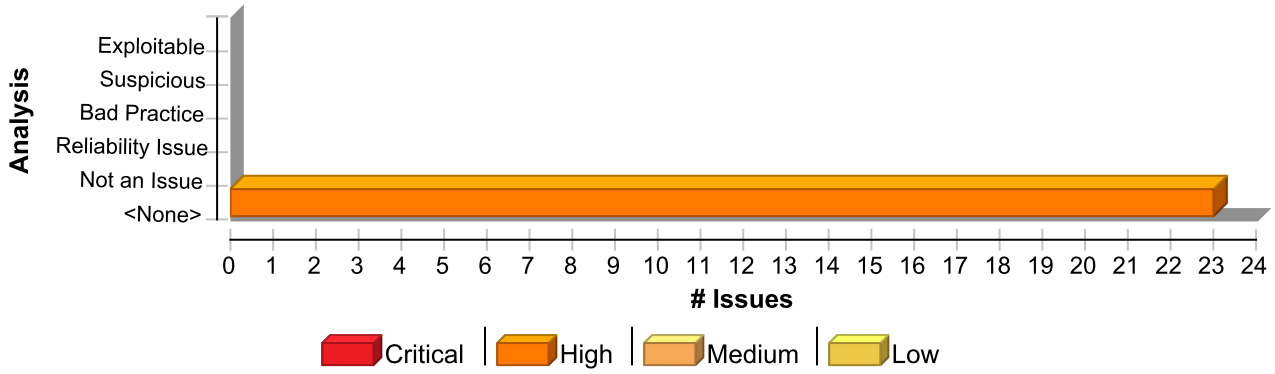
Because resource leaks can be hard to track down, establish a set of resource management patterns and idioms for your software and do not tolerate deviations from your conventions. One good pattern for addressing the error handling mistake in this example is to release the lock in a `finally` block. **Example 2:** The following code will always release the lock.

```
  ReentrantLock  myLock = new ReentrantLock();

  try {
    myLock.lock();
    performOperationInCriticalSection();
    myLock.unlock();
  }
  finally {
    if (myLock != null) {
      myLock.unlock();
    }
  }
```

Example 3: If using Kotlin, it is advisable to use the `withLock` function, removing the possibility of forgetting to unlock.

```
  val myLock = ReentrantLock()
  myLock.withLock {
    performOperationInCriticalSection()
  }
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Synchronization	23	0	0	23
Total	**23**	**0**	**0**	**23**

Unreleased Resource: Synchronization — High

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 1000 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.putLock().lockInterruptibly() : locked
Enclosing Method: enqueueWithTimeout()
File: src/main/scala/akka/dispatch/Mailbox.scala:1000
Taint Flags:

```
997  private final def enqueueWithTimeout(q: Queue[Envelope], receiver: ActorRef, envelope: Envelope): Unit = {
998  var remaining = pushTimeOut.toNanos
999
1000 putLock.lockInterruptibly()
1001 val inserted = try {
1002 var stop = false
1003 while (size.get() == capacity && !stop) {
```

src/main/scala/akka/dispatch/Mailbox.scala, line 988 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.putLock().lock() : locked

Unreleased Resource: Synchronization — High

Package: akka.dispatch

src/main/scala/akka/dispatch/Mailbox.scala, line 988 (Unreleased Resource: Synchronization) — High

Enclosing Method: signalNotFull()
File: src/main/scala/akka/dispatch/Mailbox.scala:988
Taint Flags:

```
985 }
986
987 private def signalNotFull(): Unit = {
988 putLock.lock()
989
990 try {
991 notFull.signal()
```

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 124 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: poll()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:124
Taint Flags:

```
121 }
122
123 def poll(): E = { //Tries to remove the head of the queue immediately, if fail, return null
124 lock.lock()
125 try {
126 backing.poll() match {
127 case null => null.asInstanceOf[E]
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 286 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: toArray()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:286

Unreleased Resource: Synchronization — High

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 286 (Unreleased Resource: Synchronization) — High

Taint Flags:

```
283 }
284
285 override def toArray[X](a: Array[X with AnyRef]) = {
286 lock.lock()
287 try backing.toArray[X](a)
288 finally lock.unlock()
289 }
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 60 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lockInterruptibly() : locked
Enclosing Method: take()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:60
Taint Flags:

```
57 }
58
59 def take(): E = { //Blocks until not empty
60 lock.lockInterruptibly()
61 try {
62 @tailrec def takeElement(): E = {
63 if (!backing.isEmpty()) {
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 162 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: remainingCapacity()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:162
Taint Flags:

```
159 }
160
```

Unreleased Resource: Synchronization — High

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 162 (Unreleased Resource: Synchronization) — High

```
161 def remainingCapacity(): Int = {
162 lock.lock()
163 try {
164 maxCapacity - backing.size()
165 } finally lock.unlock()
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 224 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: retainAll()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:224
Taint Flags:

```
221 }
222
223 override def retainAll(c: Collection[_]): Boolean = {
224 lock.lock()
225 try {
226 if (backing.retainAll(c)) {
227 val sz = backing.size()
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 256 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.~t~@outer.lock().lock() : locked
Enclosing Method: remove()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:256
Taint Flags:

```
253 if (last < 0) throw new IllegalStateException
254 val target = elements(last)
255 last = -1 //To avoid 2 subsequent removes without a next in between
256 lock.lock()
257 try {
```

Unreleased Resource: Synchronization — High

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 256 (Unreleased Resource: Synchronization) — High

```
258 @tailrec def removeTarget(i: Iterator[E] = backing.iterator()): Unit =
259 if (i.hasNext) {
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 169 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: size()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:169
Taint Flags:

```
166 }
167
168 def size(): Int = {
169 lock.lock()
170 try backing.size()
171 finally lock.unlock()
172 }
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 236 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: iterator()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:236
Taint Flags:

```
233 }
234
235 def iterator(): Iterator[E] = {
236 lock.lock
237 try {
238 val elements = backing.toArray
239 new Iterator[E] {
```

Unreleased Resource: Synchronization — High

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 175 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: peek()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:175
Taint Flags:

```
172 }
173
174 def peek(): E = {
175 lock.lock()
176 try backing.peek()
177 finally lock.unlock()
178 }
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 206 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: containsAll()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:206
Taint Flags:

```
203 }
204
205 override def containsAll(c: Collection[_]): Boolean = {
206 lock.lock()
207 try backing.containsAll(c)
208 finally lock.unlock()
209 }
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 92 (Unreleased Resource: Synchronization) — High

Issue Details

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

Unreleased Resource: Synchronization — High

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 92 (Unreleased Resource: Synchronization) — High

Sink Details

Sink: this.lock().lockInterruptibly() : locked
Enclosing Method: offer()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:92
Taint Flags:

```
89
90 def offer(e: E, timeout: Long, unit: TimeUnit): Boolean = { //Tries to do it within the timeout, return false if fail
91 if (e eq null) throw new NullPointerException
92 lock.lockInterruptibly()
93 try {
94 @tailrec def offerElement(remainingNanos: Long): Boolean = {
95 if (backing.size() < maxCapacity) {
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 274 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: toArray()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:274
Taint Flags:

```
271 }
272
273 override def toArray(): Array[AnyRef] = {
274 lock.lock()
275 try backing.toArray
276 finally lock.unlock()
277 }
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 212 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: removeAll()

Unreleased Resource: Synchronization — High

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 212 (Unreleased Resource: Synchronization) — High

File: src/main/scala/akka/util/BoundedBlockingQueue.scala:212
Taint Flags:

```
209 }
210
211 override def removeAll(c: Collection[_]): Boolean = {
212 lock.lock()
213 try {
214 if (backing.removeAll(c)) {
215 val sz = backing.size()
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 79 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: offer()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:79
Taint Flags:

```
76
77 def offer(e: E): Boolean = { //Tries to do it immediately, if fail return false
78 if (e eq null) throw new NullPointerException
79 lock.lock()
80 try {
81 if (backing.size() == maxCapacity) false
82 else {
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 148 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: contains()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:148
Taint Flags:

```
145
```

Unreleased Resource: Synchronization — High

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 148 (Unreleased Resource: Synchronization) — High

```
146 override def contains(e: AnyRef): Boolean = {
147 if (e eq null) throw new NullPointerException
148 lock.lock()
149 try backing.contains(e)
150 finally lock.unlock()
151 }
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 188 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: drainTo()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:188
Taint Flags:

```
185 if (c eq backing) throw new IllegalArgumentException
186 if (maxElements <= 0) 0
187 else {
188 lock.lock()
189 try {
190 @tailrec def drainOne(n: Int = 0): Int = {
191 if (n < maxElements) {
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 43 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lockInterruptibly() : locked
Enclosing Method: put()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:43
Taint Flags:

```
40
41 def put(e: E): Unit = { //Blocks until not full
42 if (e eq null) throw new NullPointerException
43 lock.lockInterruptibly()
```

Unreleased Resource: Synchronization — High

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 43 (Unreleased Resource: Synchronization) — High

```
44
45 try {
46 @tailrec def putElement(): Unit = {
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 154 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: clear()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:154
Taint Flags:

```
151 }
152
153 override def clear(): Unit = {
154 lock.lock()
155 try {
156 backing.clear()
157 notFull.signalAll()
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 280 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: isEmpty()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:280
Taint Flags:

```
277 }
278
279 override def isEmpty(): Boolean = {
280 lock.lock()
281 try backing.isEmpty()
282 finally lock.unlock()
283 }
```

Unreleased Resource: Synchronization — High

Package: akka.util

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 107 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lockInterruptibly() : locked
Enclosing Method: poll()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:107
Taint Flags:

```
104 }
105
106 def poll(timeout: Long, unit: TimeUnit): E = { //Tries to do it within the timeout, returns null if fail
107 lock.lockInterruptibly()
108 try {
109 @tailrec def pollElement(remainingNanos: Long): E = {
110 backing.poll() match {
```

src/main/scala/akka/util/BoundedBlockingQueue.scala, line 137 (Unreleased Resource: Synchronization) — High

Issue Details

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

Sink Details

Sink: this.lock().lock() : locked
Enclosing Method: remove()
File: src/main/scala/akka/util/BoundedBlockingQueue.scala:137
Taint Flags:

```
134
135 override def remove(e: AnyRef): Boolean = { //Tries to do it immediately, if fail, return false
136 if (e eq null) throw new NullPointerException
137 lock.lock()
138 try {
139 if (backing.remove(e)) {
140 notFull.signal()
```

Unsafe Reflection (1 issue)

Abstract

An attacker may be able to create unexpected control flow paths through the application, potentially bypassing security checks.

Explanation

If an attacker can supply values that the application then uses to determine which class to instantiate or which method to invoke, the potential exists for the attacker to create control flow paths through the application that were not intended by the application developers. This attack vector may allow the attacker to bypass authentication or access control checks or otherwise cause the application to behave in an unexpected manner. Even the ability to control the arguments passed to a given method or constructor may give a wily attacker the edge necessary to mount a successful attack. This situation becomes a doomsday scenario if the attacker may upload files into a location that appears on the application's classpath or add new entries to the application's classpath. Under either of these conditions, the attacker may use reflection to introduce new, presumably malicious, behavior into the application. **Example:** A common reason that programmers use the reflection API is to implement their own command dispatcher. The following example shows a command dispatcher that does not use reflection:

```
String ctl = request.getParameter("ctl");
Worker ao = null;
if (ctl.equals("Add")) {
  ao = new AddCommand();
} else if (ctl.equals("Modify")) {
  ao = new ModifyCommand();
} else {
  throw new UnknownActionError();
}
ao.doAction(request);
```

A programmer might refactor this code to use reflection as follows:

```
    String ctl = request.getParameter("ctl");
    Class cmdClass = Class.forName(ctl + "Command");
    Worker ao = (Worker) cmdClass.newInstance();
    ao.doAction(request);
```

The refactoring initially appears to offer a number of advantages. There are fewer lines of code, the `if/else` blocks have been entirely eliminated, and it is now possible to add new command types without modifying the command dispatcher. However, the refactoring allows an attacker to instantiate any object that implements the `Worker` interface. If the command dispatcher is still responsible for access control, then whenever programmers create a new class that implements the `Worker` interface, they must remember to modify the dispatcher's access control code. If they fail to modify the access control code, then some `Worker` classes will not have any access control. One way to address this access control problem is to make the `Worker` object responsible for performing the access control check. An example of the re-refactored code is as follows:

```
String ctl = request.getParameter("ctl");
Class cmdClass = Class.forName(ctl + "Command");
Worker ao = (Worker) cmdClass.newInstance();
<b>ao.checkAccessControl(request);</b>
ao.doAction(request);
```

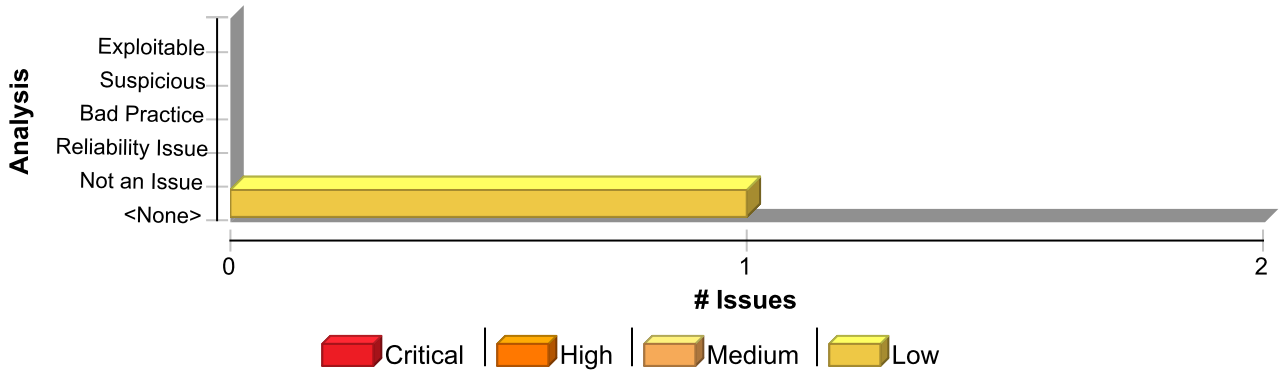
Although this is an improvement, it encourages a decentralized approach to access control, which makes it easier for programmers to make access control mistakes. This code also highlights another security problem with using reflection to build a command dispatcher. An attacker may invoke the default constructor for any kind of object. In fact, the attacker is not even constrained to objects that implement the `Worker` interface; the default constructor for any object in the system can be invoked. If the object does not implement the `Worker` interface, a `ClassCastException` will be thrown before the assignment to `ao`, but if the constructor performs operations that work in the attacker's favor, the damage will have already been done. Although this scenario is relatively benign in simple applications, in larger applications where complexity grows exponentially it is not unreasonable to assume that

an attacker could find a constructor to leverage as part of an attack. Access checks may also be compromised further down the code execution chain, if certain Java APIs that perform tasks using the immediate caller's class loader check, are invoked on untrusted objects returned by reflection calls. These Java APIs bypass the SecurityManager check that ensures all callers in the execution chain have the requisite security permissions. Care should be taken to ensure these APIs are not invoked on the untrusted objects returned by reflection as they can bypass security access checks and leave the system vulnerable to remote attacks. For more information on these Java APIs please refer to Guideline 9 of The Secure Coding Guidelines for the Java Programming Language.

Recommendation

The best way to prevent unsafe reflection is with a level of indirection: create a list of legitimate names that users are allowed to specify, and only allow users to select from the list. With this approach, input provided by users is never used directly to specify a name that is passed to the reflection API. Reflection can also be used to create a custom data-driven architecture, whereby a configuration file determines the types and combinations of objects that are used by the application. This style of programming introduces the following security concerns: - The configuration file that controls the program is an essential part of the program's source code and must be protected and reviewed accordingly. - Because the configuration file is unique to the application, unique work must be performed to evaluate the security of the design. - Because the semantics of the application are now governed by a configuration file with a custom format, custom rules are required for obtaining optimal static analysis results. For these reasons, avoid using this style of design unless your team can devote a large amount of effort to security evaluation.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unsafe Reflection	1	0	0	1
Total	**1**	**0**	**0**	**1**

Unsafe Reflection	**Low**
Package: src.main.scala.akka.actor	
src/main/scala/akka/actor/ReflectiveDynamicAccess.scala, line 29 (Unsafe Reflection)	**Low**

Issue Details

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

Source Details

Source: main(0)
File: src/main/scala/akka/Main.scala:23

Unsafe Reflection	Low
Package: src.main.scala.akka.actor	
src/main/scala/akka/actor/ReflectiveDynamicAccess.scala, line 29 (Unsafe Reflection)	**Low**

```
20  * the actor system when the top level actor is terminated.
21  */
22 @deprecated("Implement your own main class instead, from which you start the ActorSystem and actors.", "2.6.0")
23 object Main {
24
25  /**
26  * @param args one argument: the class of the application supervisor actor
```

Sink Details

Sink: java.lang.Class.forName()
Enclosing Method: apply()
File: src/main/scala/akka/actor/ReflectiveDynamicAccess.scala:29
Taint Flags: ARGS

```
26
27  override def getClassFor[T: ClassTag](fqcn: String): Try[Class[_ <: T]] =
28  Try[Class[_ <: T]]({
29  val c = Class.forName(fqcn, false, classLoader).asInstanceOf[Class[_ <: T]]
30  val t = implicitly[ClassTag[T]].runtimeClass
31  if (t.isAssignableFrom(c)) c else throw new ClassCastException(t.toString + " is not assignable from " + c)
32  })
```