Fortify Standalone Report Generator

Developer Workbook

akka-bench-jmh

Table of Contents

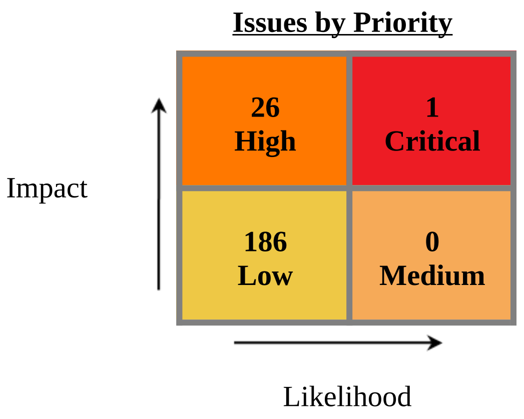
Executive Summary
Project Description
Issue Breakdown by Fortify Categories
Results Outline

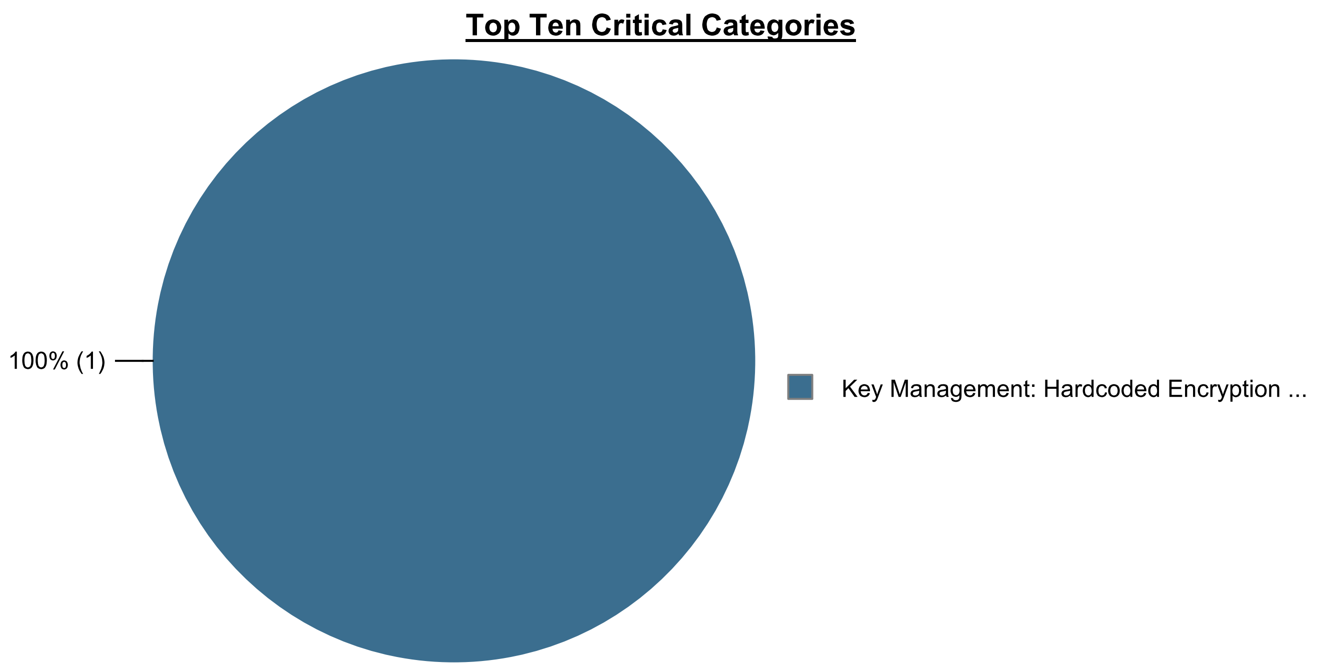
Executive Summary

This workbook is intended to provide all necessary details and information for a developer to understand and remediate the different issues discovered during the akka-bench-jmh 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-bench-jmh
Project Version:	
SCA:	Results Present
WebInspect:	Results Not Present
WebInspect Agent:	Results Not Present
Other:	Results Not Present





Project Description

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

SCA

Date of Last Analysis:	Jun 16, 2022, 11:16 AM	**Engine Version:**	21.1.1.0009
Host Name:	Jacks-Work-MBP.local	**Certification:**	VALID
Number of Files:	76	**Lines of Code:**	2,657

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

Issue Breakdown by Fortify Categories

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

Category	Fortify Priority (audited/total)				Total Issues
	Critical	High	Medium	Low	
Code Correctness: Call to System.gc()	0	0	0	0 / 5	0 / 5
Code Correctness: Constructor Invokes Overridable Function	0	0	0	0 / 150	0 / 150
Code Correctness: Erroneous String Compare	0	0	0	0 / 8	0 / 8
Code Correctness: Misleading Method Signature	0	0	0	0 / 1	0 / 1
Code Correctness: Non-Static Inner Class Implements Serializable	0	0	0	0 / 7	0 / 7
Code Correctness: toString on Array	0	0	0	0 / 1	0 / 1
Dead Code: Expression is Always false	0	0	0	0 / 1	0 / 1
Insecure Randomness	0	0 / 21	0	0	0 / 21
Insecure Randomness: Hardcoded Seed	0	0 / 1	0	0	0 / 1
J2EE Bad Practices: Leftover Debug Code	0	0	0	0 / 1	0 / 1
J2EE Bad Practices: Threads	0	0	0	0 / 5	0 / 5
Key Management: Hardcoded Encryption Key	0 / 1	0	0	0	0 / 1
Poor Style: Value Never Read	0	0	0	0 / 5	0 / 5
System Information Leak	0	0	0	0 / 1	0 / 1
System Information Leak: Internal	0	0	0	0 / 1	0 / 1
Unreleased Resource: Streams	0	0 / 1	0	0	0 / 1
Weak SecurityManager Check: Overridable Method	0	0 / 3	0	0	0 / 3

Results Outline

Code Correctness: Call to System.gc() (5 issues)

Abstract

Explicit requests for garbage collection are a bellwether indicating likely performance problems.

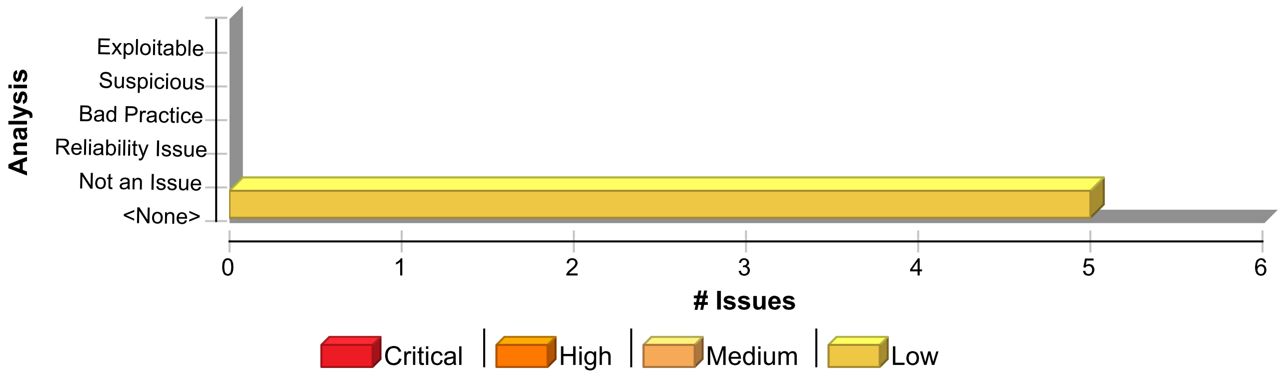
Explanation

At some point in every Java developer's career, a problem surfaces that appears to be so mysterious, impenetrable, and impervious to debugging that there seems to be no alternative but to blame the garbage collector. Especially when the bug is related to time and state, there may be a hint of empirical evidence to support this theory: inserting a call to `System.gc()` sometimes seems to make the problem go away. In almost every case we have seen, calling `System.gc()` is the wrong thing to do. In fact, calling `System.gc()` can cause performance problems if it is invoked too often.

Recommendation

When it seems as though calling `System.gc()` has solved a problem, look for other explanations, particularly ones that involve time and interaction between threads, processes, or the JVM and the operating system. I/O buffering, synchronization, and race conditions are all likely culprits.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Call to System.gc()	5	0	0	5
Total	**5**	**0**	**0**	**5**

Code Correctness: Call to System.gc() **Low**

Package: akka.actor

actor/TellOnlyBenchmark.scala, line 74 (Code Correctness: Call to System.gc()) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: Call to System.gc() — Low

Package: akka.actor

actor/TellOnlyBenchmark.scala, line 74 (Code Correctness: Call to System.gc()) — Low

Sink Details

Sink: FunctionCall: gc
Enclosing Method: setupIteration()
File: actor/TellOnlyBenchmark.scala:74
Taint Flags:

```
71 probe.expectMsg(message)
72 probe.send(actor, flipDrop)
73 probe.expectNoMessage(200.millis)
74 System.gc()
75 }
76
77 @TearDown(Level.Iteration)
```

Package: akka.dispatch

dispatch/NodeQueueBenchmark.scala, line 61 (Code Correctness: Call to System.gc()) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: gc
Enclosing Method: waitInBetween()
File: dispatch/NodeQueueBenchmark.scala:61
Taint Flags:

```
58 val probe = TestProbe()
59 probe.send(ref, Stop)
60 probe.expectMsg(Stop)
61 System.gc()
62 System.gc()
63 System.gc()
64 }
```

dispatch/NodeQueueBenchmark.scala, line 62 (Code Correctness: Call to System.gc()) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: gc
Enclosing Method: waitInBetween()
File: dispatch/NodeQueueBenchmark.scala:62
Taint Flags:

Code Correctness: Call to System.gc() — Low

Package: akka.dispatch

dispatch/NodeQueueBenchmark.scala, line 62 (Code Correctness: Call to System.gc()) — Low

```
59 probe.send(ref, Stop)
60 probe.expectMsg(Stop)
61 System.gc()
62 System.gc()
63 System.gc()
64 }
65
```

dispatch/NodeQueueBenchmark.scala, line 63 (Code Correctness: Call to System.gc()) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: gc
Enclosing Method: waitInBetween()
File: dispatch/NodeQueueBenchmark.scala:63
Taint Flags:

```
60 probe.expectMsg(Stop)
61 System.gc()
62 System.gc()
63 System.gc()
64 }
65
66 @Benchmark
```

Package: akka.remote.artery

remote/artery/CodecBenchmark.scala, line 195 (Code Correctness: Call to System.gc()) — Low

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: gc
Enclosing Method: setupIteration()
File: remote/artery/CodecBenchmark.scala:195
Taint Flags:

```
192
193 @Setup(Level.Iteration)
194 def setupIteration(): Unit = {
195 System.gc()
196 }
```

Code Correctness: Call to System.gc()	Low

Package: akka.remote.artery

remote/artery/CodecBenchmark.scala, line 195 (Code Correctness: Call to System.gc())	Low

```
197
198  @TearDown(Level.Iteration)
```

Code Correctness: Constructor Invokes Overridable Function (150 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.

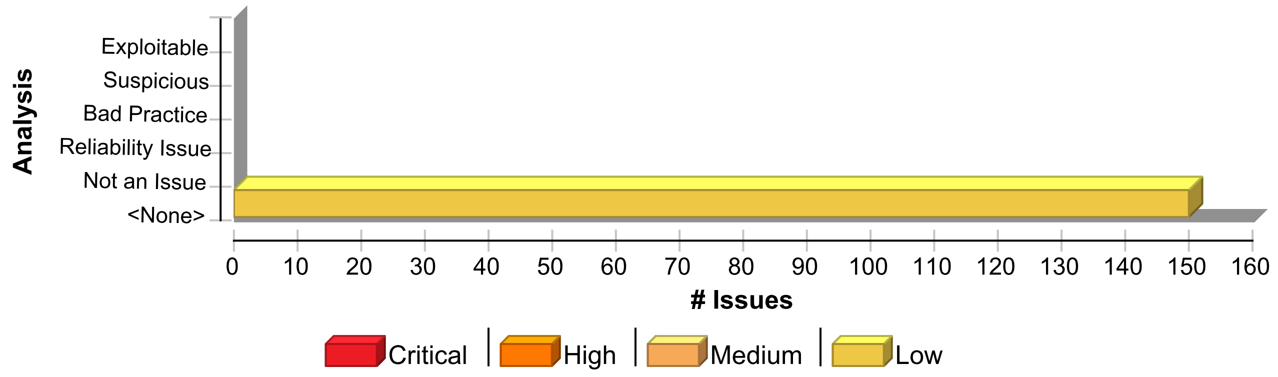
Recommendation

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	150	0	0	150
Total	**150**	**0**	**0**	**150**

Code Correctness: Constructor Invokes Overridable Function	**Low**

Package: akka.actor

actor/DirectByteBufferPoolBenchmark.scala, line 51 (Code Correctness: Constructor Invokes Overridable Function)	**Low**

Issue Details

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

actor/DirectByteBufferPoolBenchmark.scala, line 51 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: MAX_LIVE_BUFFERS
Enclosing Method: DirectByteBufferPoolBenchmark()
File: actor/DirectByteBufferPoolBenchmark.scala:51
Taint Flags:

```
48 }
49 }
50
51 private val unpooledHeapBuffers = new Array[ByteBuffer](MAX_LIVE_BUFFERS)
52
53 private val pooledDirectBuffers = new Array[ByteBuffer](MAX_LIVE_BUFFERS)
54 private val unpooledDirectBuffers = new Array[ByteBuffer](MAX_LIVE_BUFFERS)
```

actor/DirectByteBufferPoolBenchmark.scala, line 53 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: MAX_LIVE_BUFFERS
Enclosing Method: DirectByteBufferPoolBenchmark()
File: actor/DirectByteBufferPoolBenchmark.scala:53
Taint Flags:

```
50
51 private val unpooledHeapBuffers = new Array[ByteBuffer](MAX_LIVE_BUFFERS)
52
53 private val pooledDirectBuffers = new Array[ByteBuffer](MAX_LIVE_BUFFERS)
54 private val unpooledDirectBuffers = new Array[ByteBuffer](MAX_LIVE_BUFFERS)
55
56 import org.openjdk.jmh.annotations.Benchmark
```

actor/DirectByteBufferPoolBenchmark.scala, line 54 (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

actor/DirectByteBufferPoolBenchmark.scala, line 54 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: MAX_LIVE_BUFFERS
Enclosing Method: DirectByteBufferPoolBenchmark()
File: actor/DirectByteBufferPoolBenchmark.scala:54
Taint Flags:

```
51  private val unpooledHeapBuffers = new Array[ByteBuffer](MAX_LIVE_BUFFERS)
52
53  private val pooledDirectBuffers = new Array[ByteBuffer](MAX_LIVE_BUFFERS)
54  private val unpooledDirectBuffers = new Array[ByteBuffer](MAX_LIVE_BUFFERS)
55
56  import org.openjdk.jmh.annotations.Benchmark
57
```

actor/RouterPoolCreationBenchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: system
Enclosing Method: RouterPoolCreationBenchmark()
File: actor/RouterPoolCreationBenchmark.scala:25
Taint Flags:

```
22 @Measurement(iterations = 100)
23 class RouterPoolCreationBenchmark {
24  implicit val system: ActorSystem = ActorSystem()
25  val probe = TestProbe()
26
27  Props[TestActors.EchoActor]()
28
```

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

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: system
Enclosing Method: StashCreationBenchmark()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

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

File: actor/StashCreationBenchmark.scala:39
Taint Flags:

```
36  }
37  """)
38  implicit val system: ActorSystem = ActorSystem("StashCreationBenchmark", conf)
39  val probe = TestProbe()
40
41  @TearDown(Level.Trial)
42  def shutdown(): Unit = {
```

actor/ScheduleBenchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: system
Enclosing Method: ScheduleBenchmark()
File: actor/ScheduleBenchmark.scala:25
Taint Flags:

```
22 @Measurement(iterations = 20, time = 1700, timeUnit = TimeUnit.MILLISECONDS)
23 class ScheduleBenchmark {
24  implicit val system: ActorSystem = ActorSystem()
25  val scheduler: Scheduler = system.scheduler
26  val interval: FiniteDuration = 25.millis
27  val within: FiniteDuration = 2.seconds
28  implicit val timeout: Timeout = Timeout(within)
```

actor/ScheduleBenchmark.scala, line 28 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: within
Enclosing Method: ScheduleBenchmark()
File: actor/ScheduleBenchmark.scala:28
Taint Flags:

```
25  val scheduler: Scheduler = system.scheduler
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor

actor/ScheduleBenchmark.scala, line 28 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
val interval: FiniteDuration = 25.millis
val within: FiniteDuration = 2.seconds
implicit val timeout: Timeout = Timeout(within)

@Param(Array("4", "16", "64"))
var to = 0
```

actor/StashCreationBenchmark.scala, line 38 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: conf
Enclosing Method: StashCreationBenchmark()
File: actor/StashCreationBenchmark.scala:38
Taint Flags:

```
stash-capacity = 1000
}
""")
implicit val system: ActorSystem = ActorSystem("StashCreationBenchmark", conf)
val probe = TestProbe()

@TearDown(Level.Trial)
```

Package: akka.actor.typed

actor/typed/TypedActorBenchmark.scala, line 51 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: timeout
Enclosing Method: TypedActorBenchmark()
File: actor/typed/TypedActorBenchmark.scala:51
Taint Flags:

```

implicit var system: ActorSystem[Start] = _

```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.actor.typed

actor/typed/TypedActorBenchmark.scala, line 51 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
implicit val askTimeout: akka.util.Timeout = akka.util.Timeout(timeout)

@Setup(Level.Trial)
def setup(): Unit = {
```

Package: akka.actor.typed.delivery

actor/typed/delivery/ReliableDeliveryBenchmark.scala, line 213 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: timeout
Enclosing Method: ReliableDeliveryBenchmark()
File: actor/typed/delivery/ReliableDeliveryBenchmark.scala:213
Taint Flags:

```

implicit var system: ActorSystem[Guardian.Command] = _

implicit val askTimeout: akka.util.Timeout = akka.util.Timeout(timeout)

@Setup(Level.Trial)
def setup(): Unit = {
```

Package: akka.cluster.ddata

cluster/ddata/ORSetMergeBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeB
Enclosing Method: ORSetMergeBenchmark()
File: cluster/ddata/ORSetMergeBenchmark.scala:41
Taint Flags:

```
val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/ORSetMergeBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
val nodesIndex = Iterator.from(0)
def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)

```

cluster/ddata/VersionVectorBenchmark.scala, line 37 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: VersionVectorBenchmark()
File: cluster/ddata/VersionVectorBenchmark.scala:37
Taint Flags:

```
var size = 0

val nodeA = UniqueAddress(Address("akka", "Sys", "aaaa", 2552), 1L)
val nodeB = UniqueAddress(nodeA.address.copy(host = Some("bbbb")), 2L)
val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
```

cluster/ddata/VersionVectorBenchmark.scala, line 38 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: VersionVectorBenchmark()
File: cluster/ddata/VersionVectorBenchmark.scala:38
Taint Flags:

```

val nodeA = UniqueAddress(Address("akka", "Sys", "aaaa", 2552), 1L)
val nodeB = UniqueAddress(nodeA.address.copy(host = Some("bbbb")), 2L)
val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/VersionVectorBenchmark.scala, line 38 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
```

cluster/ddata/VersionVectorBenchmark.scala, line 39 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: VersionVectorBenchmark()
File: cluster/ddata/VersionVectorBenchmark.scala:39
Taint Flags:

```
36 val nodeA = UniqueAddress(Address("akka", "Sys", "aaaa", 2552), 1L)
37 val nodeB = UniqueAddress(nodeA.address.copy(host = Some("bbbb")), 2L)
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
```

cluster/ddata/VersionVectorBenchmark.scala, line 40 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: VersionVectorBenchmark()
File: cluster/ddata/VersionVectorBenchmark.scala:40
Taint Flags:

```
37 val nodeB = UniqueAddress(nodeA.address.copy(host = Some("bbbb")), 2L)
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
43 def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/VersionVectorBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: VersionVectorBenchmark()
File: cluster/ddata/VersionVectorBenchmark.scala:41
Taint Flags:

```
38  val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39  val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40  val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41  val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42  val nodesIndex = Iterator.from(0)
43  def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
44
```

cluster/ddata/ORSetMergeBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeC
Enclosing Method: ORSetMergeBenchmark()
File: cluster/ddata/ORSetMergeBenchmark.scala:41
Taint Flags:

```
38  val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39  val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40  val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41  val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42  val nodesIndex = Iterator.from(0)
43  def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
44
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 53 (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.cluster.ddata

cluster/ddata/ORSetSerializationBenchmark.scala, line 53 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: system2
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:53
Taint Flags:

```
50
51 private val orSet = {
52 val selfUniqueAddress1 = SelfUniqueAddress(Cluster(system1).selfUniqueAddress)
53 val selfUniqueAddress2 = SelfUniqueAddress(Cluster(system2).selfUniqueAddress)
54 val set1 = ref1.foldLeft(ORSet.empty[ActorRef]) { case (acc, r) => acc.add(selfUniqueAddress1, r) }
55 val set2 = ref2.foldLeft(ORSet.empty[ActorRef]) { case (acc, r) => acc.add(selfUniqueAddress2, r) }
56 set1.merge(set2)
```

cluster/ddata/VersionVectorBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeC
Enclosing Method: VersionVectorBenchmark()
File: cluster/ddata/VersionVectorBenchmark.scala:41
Taint Flags:

```
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
43 def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
44
```

cluster/ddata/VersionVectorBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeD
Enclosing Method: VersionVectorBenchmark()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/VersionVectorBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: cluster/ddata/VersionVectorBenchmark.scala:41
Taint Flags:

```
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
43 def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
44
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 55 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: ref2
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:55
Taint Flags:

```
52 val selfUniqueAddress1 = SelfUniqueAddress(Cluster(system1).selfUniqueAddress)
53 val selfUniqueAddress2 = SelfUniqueAddress(Cluster(system2).selfUniqueAddress)
54 val set1 = ref1.foldLeft(ORSet.empty[ActorRef]) { case (acc, r) => acc.add(selfUniqueAddress1, r) }
55 val set2 = ref2.foldLeft(ORSet.empty[ActorRef]) { case (acc, r) => acc.add(selfUniqueAddress2, r) }
56 set1.merge(set2)
57 }
58
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 54 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: ref1
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:54
Taint Flags:

```
51 private val orSet = {
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/ORSetSerializationBenchmark.scala, line 54 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
52 val selfUniqueAddress1 = SelfUniqueAddress(Cluster(system1).selfUniqueAddress)
53 val selfUniqueAddress2 = SelfUniqueAddress(Cluster(system2).selfUniqueAddress)
54 val set1 = ref1.foldLeft(ORSet.empty[ActorRef]) { case (acc, r) => acc.add(selfUniqueAddress1, r) }
55 val set2 = ref2.foldLeft(ORSet.empty[ActorRef]) { case (acc, r) => acc.add(selfUniqueAddress2, r) }
56 set1.merge(set2)
57 }
```

cluster/ddata/ORSetMergeBenchmark.scala, line 37 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: ORSetMergeBenchmark()
File: cluster/ddata/ORSetMergeBenchmark.scala:37
Taint Flags:

```
34 var set1Size = 0
35
36 val nodeA = UniqueAddress(Address("akka", "Sys", "aaaa", 2552), 1L)
37 val nodeB = UniqueAddress(nodeA.address.copy(host = Some("bbbb")), 2L)
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
```

cluster/ddata/ORSetMergeBenchmark.scala, line 38 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: ORSetMergeBenchmark()
File: cluster/ddata/ORSetMergeBenchmark.scala:38
Taint Flags:

```
35
36 val nodeA = UniqueAddress(Address("akka", "Sys", "aaaa", 2552), 1L)
37 val nodeB = UniqueAddress(nodeA.address.copy(host = Some("bbbb")), 2L)
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/ORSetMergeBenchmark.scala, line 38 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
```

cluster/ddata/ORSetMergeBenchmark.scala, line 39 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: ORSetMergeBenchmark()
File: cluster/ddata/ORSetMergeBenchmark.scala:39
Taint Flags:

```
36 val nodeA = UniqueAddress(Address("akka", "Sys", "aaaa", 2552), 1L)
37 val nodeB = UniqueAddress(nodeA.address.copy(host = Some("bbbb")), 2L)
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
```

cluster/ddata/ORSetMergeBenchmark.scala, line 40 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: ORSetMergeBenchmark()
File: cluster/ddata/ORSetMergeBenchmark.scala:40
Taint Flags:

```
37 val nodeB = UniqueAddress(nodeA.address.copy(host = Some("bbbb")), 2L)
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
43 def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/ORSetMergeBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeA
Enclosing Method: ORSetMergeBenchmark()
File: cluster/ddata/ORSetMergeBenchmark.scala:41
Taint Flags:

```
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
43 def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
44
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 52 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: system1
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:52
Taint Flags:

```
49 private val ref2 = (1 to 10).map(n => system2.actorOf(Props.empty, s"ref2-$n"))
50
51 private val orSet = {
52 val selfUniqueAddress1 = SelfUniqueAddress(Cluster(system1).selfUniqueAddress)
53 val selfUniqueAddress2 = SelfUniqueAddress(Cluster(system2).selfUniqueAddress)
54 val set1 = ref1.foldLeft(ORSet.empty[ActorRef]) { case (acc, r) => acc.add(selfUniqueAddress1, r) }
55 val set2 = ref2.foldLeft(ORSet.empty[ActorRef]) { case (acc, r) => acc.add(selfUniqueAddress2, r) }
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 59 (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.cluster.ddata

cluster/ddata/ORSetSerializationBenchmark.scala, line 59 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: system1
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:59
Taint Flags:

```
56 set1.merge(set2)
57 }
58
59 private val serialization = SerializationExtension(system1)
60 private val serializerId = serialization.findSerializerFor(orSet).identifier
61 private val manifest = Serializers.manifestFor(serialization.findSerializerFor(orSet), orSet)
62
```

cluster/ddata/ORSetMergeBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeE
Enclosing Method: ORSetMergeBenchmark()
File: cluster/ddata/ORSetMergeBenchmark.scala:41
Taint Flags:

```
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
43 def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
44
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 45 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: ORSetSerializationBenchmark()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/ORSetSerializationBenchmark.scala, line 45 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: cluster/ddata/ORSetSerializationBenchmark.scala:45
Taint Flags:

```
akka.remote.artery.canonical.port = 0
""")

private val system1 = ActorSystem("ORSetSerializationBenchmark", config)
private val system2 = ActorSystem("ORSetSerializationBenchmark", config)

private val ref1 = (1 to 10).map(n => system1.actorOf(Props.empty, s"ref1-$n"))
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 46 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:46
Taint Flags:

```
""")

private val system1 = ActorSystem("ORSetSerializationBenchmark", config)
private val system2 = ActorSystem("ORSetSerializationBenchmark", config)

private val ref1 = (1 to 10).map(n => system1.actorOf(Props.empty, s"ref1-$n"))
private val ref2 = (1 to 10).map(n => system2.actorOf(Props.empty, s"ref2-$n"))
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 60 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: orSet
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:60
Taint Flags:

```
}
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/ORSetSerializationBenchmark.scala, line 60 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
58
59 private val serialization = SerializationExtension(system1)
60 private val serializerId = serialization.findSerializerFor(orSet).identifier
61 private val manifest = Serializers.manifestFor(serialization.findSerializerFor(orSet), orSet)
62
63 @TearDown
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 61 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: orSet
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:61
Taint Flags:

```
58
59 private val serialization = SerializationExtension(system1)
60 private val serializerId = serialization.findSerializerFor(orSet).identifier
61 private val manifest = Serializers.manifestFor(serialization.findSerializerFor(orSet), orSet)
62
63 @TearDown
64 def shutdown(): Unit = {
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 61 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: orSet
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:61
Taint Flags:

```
58
59 private val serialization = SerializationExtension(system1)
60 private val serializerId = serialization.findSerializerFor(orSet).identifier
61 private val manifest = Serializers.manifestFor(serialization.findSerializerFor(orSet), orSet)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/ORSetSerializationBenchmark.scala, line 61 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
62
63 @TearDown
64 def shutdown(): Unit = {
```

cluster/ddata/VersionVectorBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeE
Enclosing Method: VersionVectorBenchmark()
File: cluster/ddata/VersionVectorBenchmark.scala:41
Taint Flags:

```
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
43 def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
44
```

cluster/ddata/ORSetSerializationBenchmark.scala, line 60 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serialization
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:60
Taint Flags:

```
57 }
58
59 private val serialization = SerializationExtension(system1)
60 private val serializerId = serialization.findSerializerFor(orSet).identifier
61 private val manifest = Serializers.manifestFor(serialization.findSerializerFor(orSet), orSet)
62
63 @TearDown
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata

cluster/ddata/ORSetSerializationBenchmark.scala, line 61 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serialization
Enclosing Method: ORSetSerializationBenchmark()
File: cluster/ddata/ORSetSerializationBenchmark.scala:61
Taint Flags:

```
58
59  private val serialization = SerializationExtension(system1)
60  private val serializerId = serialization.findSerializerFor(orSet).identifier
61  private val manifest = Serializers.manifestFor(serialization.findSerializerFor(orSet), orSet)
62
63  @TearDown
64  def shutdown(): Unit = {
```

cluster/ddata/VersionVectorBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: nodeB
Enclosing Method: VersionVectorBenchmark()
File: cluster/ddata/VersionVectorBenchmark.scala:41
Taint Flags:

```
38  val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39  val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40  val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41  val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42  val nodesIndex = Iterator.from(0)
43  def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
44
```

cluster/ddata/ORSetMergeBenchmark.scala, line 41 (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.cluster.ddata

cluster/ddata/ORSetMergeBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: nodeD
Enclosing Method: ORSetMergeBenchmark()
File: cluster/ddata/ORSetMergeBenchmark.scala:41
Taint Flags:

```
38 val nodeC = UniqueAddress(nodeA.address.copy(host = Some("cccc")), 3L)
39 val nodeD = UniqueAddress(nodeA.address.copy(host = Some("dddd")), 4L)
40 val nodeE = UniqueAddress(nodeA.address.copy(host = Some("eeee")), 5L)
41 val nodes = Vector(nodeA, nodeB, nodeC, nodeD, nodeE)
42 val nodesIndex = Iterator.from(0)
43 def nextNode(): UniqueAddress = nodes(nodesIndex.next() % nodes.size)
44
```

Package: akka.dispatch

dispatch/CachingConfigBenchmark.scala, line 22 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: deepConfigString
Enclosing Method: CachingConfigBenchmark()
File: dispatch/CachingConfigBenchmark.scala:22
Taint Flags:

```
19
20 val deepKey = "akka.actor.deep.settings.something"
21 val deepConfigString = s"""$deepKey = something"""
22 val deepConfig = ConfigFactory.parseString(deepConfigString)
23 val deepCaching = new CachingConfig(deepConfig)
24
25 @Benchmark def deep_config = deepConfig.hasPath(deepKey)
```

dispatch/NodeQueueBenchmark.scala, line 46 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: sys

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.dispatch

dispatch/NodeQueueBenchmark.scala, line 46 (Code Correctness: Constructor Invokes Overridable Function) — Low

Enclosing Method: NodeQueueBenchmark()
File: dispatch/NodeQueueBenchmark.scala:46
Taint Flags:

```
43 }
44 """).withFallback(ConfigFactory.load())
45  implicit val sys: ActorSystem = ActorSystem("ANQ", config)
46  val ref = sys.actorOf(Props(new Actor {
47  def receive = {
48  case Stop => sender() ! Stop
49  case _ =>
```

dispatch/CachingConfigBenchmark.scala, line 21 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: deepKey
Enclosing Method: CachingConfigBenchmark()
File: dispatch/CachingConfigBenchmark.scala:21
Taint Flags:

```
18 class CachingConfigBenchmark {
19
20  val deepKey = "akka.actor.deep.settings.something"
21  val deepConfigString = s"""$deepKey = something"""
22  val deepConfig = ConfigFactory.parseString(deepConfigString)
23  val deepCaching = new CachingConfig(deepConfig)
24
```

dispatch/NodeQueueBenchmark.scala, line 45 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: NodeQueueBenchmark()
File: dispatch/NodeQueueBenchmark.scala:45
Taint Flags:

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.dispatch

dispatch/NodeQueueBenchmark.scala, line 45 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
mailbox-capacity = 1000000
}
""").withFallback(ConfigFactory.load())
implicit val sys: ActorSystem = ActorSystem("ANQ", config)
val ref = sys.actorOf(Props(new Actor {
def receive = {
case Stop => sender() ! Stop
```

dispatch/CachingConfigBenchmark.scala, line 23 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: deepConfig
Enclosing Method: CachingConfigBenchmark()
File: dispatch/CachingConfigBenchmark.scala:23
Taint Flags:

```
val deepKey = "akka.actor.deep.settings.something"
val deepConfigString = s"""$deepKey = something"""
val deepConfig = ConfigFactory.parseString(deepConfigString)
val deepCaching = new CachingConfig(deepConfig)

@Benchmark def deep_config = deepConfig.hasPath(deepKey)
@Benchmark def deep_caching = deepCaching.hasPath(deepKey)
```

Package: akka.event

event/LogLevelAccessBenchmark.scala, line 36 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: NoopBus
Enclosing Method: LogLevelAccessBenchmark()
File: event/LogLevelAccessBenchmark.scala:36
Taint Flags:

```
override def unsubscribe(subscriber: Subscriber): Unit = ()
}
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.event

event/LogLevelAccessBenchmark.scala, line 36 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
var log: BusLogging = akka.event.Logging(NoopBus, "").asInstanceOf[BusLogging]

@Benchmark
@GroupThreads(20)
```

Package: akka.remote.artery

remote/artery/LatchSink.scala, line 19 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: in
Enclosing Method: LatchSink()
File: remote/artery/LatchSink.scala:19
Taint Flags:

```

class LatchSink(countDownAfter: Int, latch: CountDownLatch) extends GraphStage[SinkShape[Any]] {
val in: Inlet[Any] = Inlet("LatchSink")
override val shape: SinkShape[Any] = SinkShape(in)

override def createLogic(inheritedAttributes: Attributes): GraphStageLogic =
new GraphStageLogic(shape) with InHandler {
```

remote/artery/SendQueueBenchmark.scala, line 35 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: SendQueueBenchmark()
File: remote/artery/SendQueueBenchmark.scala:35
Taint Flags:

```
val config = ConfigFactory.parseString("""
""")

implicit val system: ActorSystem = ActorSystem("SendQueueBenchmark", config)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.artery

remote/artery/SendQueueBenchmark.scala, line 35 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
36
37  @Setup
38  def setup(): Unit = {
```

remote/artery/CodecBenchmark.scala, line 59 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: DummyMessageInstance
Enclosing Method: CodecBenchmark()
File: remote/artery/CodecBenchmark.scala:59
Taint Flags:

```
56  val envelopeTemplateBuffer = ByteBuffer.allocate(1024 * 1024).order(ByteOrder.LITTLE_ENDIAN)
57
58  var uniqueLocalAddress: UniqueAddress = _
59  val payload = DummyMessageInstance
60
61  private val inboundContext: InboundContext = new InboundContext {
62  override def localAddress: UniqueAddress = uniqueLocalAddress
```

remote/artery/BenchTestSource.scala, line 47 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: out
Enclosing Method: BenchTestSourceSameElement()
File: remote/artery/BenchTestSource.scala:47
Taint Flags:

```
44 class BenchTestSourceSameElement[T](elements: Int, elem: T) extends GraphStage[SourceShape[T]] {
45
46  val out: Outlet[T] = Outlet("BenchTestSourceSameElement")
47  override val shape: SourceShape[T] = SourceShape(out)
48
49  override def createLogic(inheritedAttributes: Attributes): GraphStageLogic =
50  new GraphStageLogic(shape) with OutHandler {
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.artery

remote/artery/LatchSink.scala, line 48 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: in
Enclosing Method: BarrierSink()
File: remote/artery/LatchSink.scala:48
Taint Flags:

```
45 class BarrierSink(countDownAfter: Int, latch: CountDownLatch, barrierAfter: Int, barrier: CyclicBarrier)
46  extends GraphStage[SinkShape[Any]] {
47  val in: Inlet[Any] = Inlet("BarrierSink")
48  override val shape: SinkShape[Any] = SinkShape(in)
49
50  override def createLogic(inheritedAttributes: Attributes): GraphStageLogic =
51  new GraphStageLogic(shape) with InHandler {
```

remote/artery/BenchTestSource.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: out
Enclosing Method: BenchTestSource()
File: remote/artery/BenchTestSource.scala:25
Taint Flags:

```
22  (1 to elementCount).foreach(n => elements(n - 1) = n)
23
24  val out: Outlet[java.lang.Integer] = Outlet("BenchTestSource")
25  override val shape: SourceShape[java.lang.Integer] = SourceShape(out)
26
27  override def createLogic(inheritedAttributes: Attributes): GraphStageLogic =
28  new GraphStageLogic(shape) with OutHandler {
```

Package: akka.remote.artery.compress

remote/artery/compress/CountMinSketchBenchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.artery.compress

remote/artery/compress/CountMinSketchBenchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: seed
Enclosing Method: CountMinSketchBenchmark()
File: remote/artery/compress/CountMinSketchBenchmark.scala:25
Taint Flags:

```
22
23  private val seed: Int = 20160726
24
25  val rand = new Random(seed)
26
27  val preallocateIds = Array.ofDim[Int](8192)
28  val preallocateValues = Array.ofDim[Long](8192)
```

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 72 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: smallMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:72
Taint Flags:

```
69  val smallMsg1 = Small("abc", 17)
70  val smallMsg2 = Small("def", 18)
71  val smallMsg3 = Small("ghi", 19)
72  val mediumMsg1 = Medium(
73  "abc",
74  "def",
75  "ghi",
```

serialization/jackson/JacksonSerializationBench.scala, line 87 (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.jackson

serialization/jackson/JacksonSerializationBench.scala, line 87 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: smallMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:87
Taint Flags:

```
84 smallMsg1,
85 smallMsg2,
86 smallMsg3)
87 val mediumMsg2 = Medium(
88 "ABC",
89 "DEF",
90 "GHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 102 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: smallMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:102
Taint Flags:

```
99 smallMsg1,
100 smallMsg2,
101 smallMsg3)
102 val mediumMsg3 = Medium(
103 "abcABC",
104 "defDEF",
105 "ghiGHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 117 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mediumMsg1
Enclosing Method: JacksonSerializationBench()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 117 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: serialization/jackson/JacksonSerializationBench.scala:117
Taint Flags:

```
114 smallMsg1,
115 smallMsg2,
116 smallMsg3)
117 val largeMsg = Large(
118 mediumMsg1,
119 mediumMsg2,
120 mediumMsg3,
```

serialization/jackson/JacksonSerializationBench.scala, line 121 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mediumMsg1
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:121
Taint Flags:

```
118 mediumMsg1,
119 mediumMsg2,
120 mediumMsg3,
121 Vector(mediumMsg1, mediumMsg2, mediumMsg3),
122 Map("a" -> mediumMsg1, "b" -> mediumMsg2, "c" -> mediumMsg3))
123
124 val timeMsg = new TimeMessage(5.seconds, LocalDateTime.of(2019, 4, 29, 23, 15, 3, 12345), Instant.now())
```

serialization/jackson/JacksonSerializationBench.scala, line 122 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mediumMsg1
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:122
Taint Flags:

```
119 mediumMsg2,
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 122 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
mediumMsg3,
Vector(mediumMsg1, mediumMsg2, mediumMsg3),
Map("a" -> mediumMsg1, "b" -> mediumMsg2, "c" -> mediumMsg3))

val timeMsg = new TimeMessage(5.seconds, LocalDateTime.of(2019, 4, 29, 23, 15, 3, 12345), Instant.now())

```

serialization/jackson/JacksonSerializationBench.scala, line 72 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: smallMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:72
Taint Flags:

```
val smallMsg1 = Small("abc", 17)
val smallMsg2 = Small("def", 18)
val smallMsg3 = Small("ghi", 19)
val mediumMsg1 = Medium(
"abc",
"def",
"ghi",
```

serialization/jackson/JacksonSerializationBench.scala, line 87 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: smallMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:87
Taint Flags:

```
smallMsg1,
smallMsg2,
smallMsg3)
val mediumMsg2 = Medium(
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 87 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
88 "ABC",
89 "DEF",
90 "GHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 102 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: smallMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:102
Taint Flags:

```
99 smallMsg1,
100 smallMsg2,
101 smallMsg3)
102 val mediumMsg3 = Medium(
103 "abcABC",
104 "defDEF",
105 "ghiGHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 178 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMediumMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:178
Taint Flags:

```
175 val jMap = new util.HashMap[String, JMedium]()
176 jMap.put("a", jMediumMsg1)
177 jMap.put("b", jMediumMsg2)
178 jMap.put("c", jMediumMsg3)
179 val jLargeMsg = new JLarge(
180 jMediumMsg1,
181 jMediumMsg2,
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 179 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMediumMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:179
Taint Flags:

```
176 jMap.put("a", jMediumMsg1)
177 jMap.put("b", jMediumMsg2)
178 jMap.put("c", jMediumMsg3)
179 val jLargeMsg = new JLarge(
180 jMediumMsg1,
181 jMediumMsg2,
182 jMediumMsg3,
```

serialization/jackson/JacksonSerializationBench.scala, line 179 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMediumMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:179
Taint Flags:

```
176 jMap.put("a", jMediumMsg1)
177 jMap.put("b", jMediumMsg2)
178 jMap.put("c", jMediumMsg3)
179 val jLargeMsg = new JLarge(
180 jMediumMsg1,
181 jMediumMsg2,
182 jMediumMsg3,
```

serialization/jackson/JacksonSerializationBench.scala, line 72 (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.jackson

serialization/jackson/JacksonSerializationBench.scala, line 72 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: smallMsg1
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:72
Taint Flags:

```
69 val smallMsg1 = Small("abc", 17)
70 val smallMsg2 = Small("def", 18)
71 val smallMsg3 = Small("ghi", 19)
72 val mediumMsg1 = Medium(
73 "abc",
74 "def",
75 "ghi",
```

serialization/jackson/JacksonSerializationBench.scala, line 87 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: smallMsg1
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:87
Taint Flags:

```
84 smallMsg1,
85 smallMsg2,
86 smallMsg3)
87 val mediumMsg2 = Medium(
88 "ABC",
89 "DEF",
90 "GHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 102 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: smallMsg1
Enclosing Method: JacksonSerializationBench()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 102 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: serialization/jackson/JacksonSerializationBench.scala:102
Taint Flags:

```
99  smallMsg1,
100  smallMsg2,
101  smallMsg3)
102  val mediumMsg3 = Medium(
103  "abcABC",
104  "defDEF",
105  "ghiGHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 176 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMap
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:176
Taint Flags:

```
173  jSmallMsg2,
174  jSmallMsg3)
175  val jMap = new util.HashMap[String, JMedium]()
176  jMap.put("a", jMediumMsg1)
177  jMap.put("b", jMediumMsg2)
178  jMap.put("c", jMediumMsg3)
179  val jLargeMsg = new JLarge(
```

serialization/jackson/JacksonSerializationBench.scala, line 177 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMap
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:177
Taint Flags:

```
174  jSmallMsg3)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 177 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
val jMap = new util.HashMap[String, JMedium]()
jMap.put("a", jMediumMsg1)
jMap.put("b", jMediumMsg2)
jMap.put("c", jMediumMsg3)
val jLargeMsg = new JLarge(
jMediumMsg1,
```

serialization/jackson/JacksonSerializationBench.scala, line 178 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMap
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:178
Taint Flags:

```
val jMap = new util.HashMap[String, JMedium]()
jMap.put("a", jMediumMsg1)
jMap.put("b", jMediumMsg2)
jMap.put("c", jMediumMsg3)
val jLargeMsg = new JLarge(
jMediumMsg1,
jMediumMsg2,
```

serialization/jackson/JacksonSerializationBench.scala, line 179 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMap
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:179
Taint Flags:

```
jMap.put("a", jMediumMsg1)
jMap.put("b", jMediumMsg2)
jMap.put("c", jMediumMsg3)
val jLargeMsg = new JLarge(
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 179 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
jMediumMsg1,
jMediumMsg2,
jMediumMsg3,
```

serialization/jackson/JacksonSerializationBench.scala, line 130 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jSmallMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:130
Taint Flags:

```
val jSmallMsg1 = new JSmall("abc", 17)
val jSmallMsg2 = new JSmall("def", 18)
val jSmallMsg3 = new JSmall("ghi", 19)
val jMediumMsg1 = new JMedium(
"abc",
"def",
"ghi",
```

serialization/jackson/JacksonSerializationBench.scala, line 145 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jSmallMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:145
Taint Flags:

```
jSmallMsg1,
jSmallMsg2,
jSmallMsg3)
val jMediumMsg2 = new JMedium(
"ABC",
"DEF",
"GHI",
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 160 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jSmallMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:160
Taint Flags:

```
157 jSmallMsg1,
158 jSmallMsg2,
159 jSmallMsg3)
160 val jMediumMsg3 = new JMedium(
161 "abcABC",
162 "defDEF",
163 "ghiGHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 177 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMediumMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:177
Taint Flags:

```
174 jSmallMsg3)
175 val jMap = new util.HashMap[String, JMedium]()
176 jMap.put("a", jMediumMsg1)
177 jMap.put("b", jMediumMsg2)
178 jMap.put("c", jMediumMsg3)
179 val jLargeMsg = new JLarge(
180 jMediumMsg1,
```

serialization/jackson/JacksonSerializationBench.scala, line 179 (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.jackson

serialization/jackson/JacksonSerializationBench.scala, line 179 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: jMediumMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:179
Taint Flags:

```
176  jMap.put("a", jMediumMsg1)
177  jMap.put("b", jMediumMsg2)
178  jMap.put("c", jMediumMsg3)
179  val jLargeMsg = new JLarge(
180  jMediumMsg1,
181  jMediumMsg2,
182  jMediumMsg3,
```

serialization/jackson/JacksonSerializationBench.scala, line 179 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMediumMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:179
Taint Flags:

```
176  jMap.put("a", jMediumMsg1)
177  jMap.put("b", jMediumMsg2)
178  jMap.put("c", jMediumMsg3)
179  val jLargeMsg = new JLarge(
180  jMediumMsg1,
181  jMediumMsg2,
182  jMediumMsg3,
```

serialization/jackson/JacksonSerializationBench.scala, line 130 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jSmallMsg1
Enclosing Method: JacksonSerializationBench()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 130 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: serialization/jackson/JacksonSerializationBench.scala:130
Taint Flags:

```
127 val jSmallMsg1 = new JSmall("abc", 17)
128 val jSmallMsg2 = new JSmall("def", 18)
129 val jSmallMsg3 = new JSmall("ghi", 19)
130 val jMediumMsg1 = new JMedium(
131 "abc",
132 "def",
133 "ghi",
```

serialization/jackson/JacksonSerializationBench.scala, line 145 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jSmallMsg1
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:145
Taint Flags:

```
142 jSmallMsg1,
143 jSmallMsg2,
144 jSmallMsg3)
145 val jMediumMsg2 = new JMedium(
146 "ABC",
147 "DEF",
148 "GHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 160 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jSmallMsg1
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:160
Taint Flags:

```
157 jSmallMsg1,
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 160 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
158 jSmallMsg2,
159 jSmallMsg3)
160 val jMediumMsg3 = new JMedium(
161 "abcABC",
162 "defDEF",
163 "ghiGHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 176 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMediumMsg1
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:176
Taint Flags:

```
173 jSmallMsg2,
174 jSmallMsg3)
175 val jMap = new util.HashMap[String, JMedium]()
176 jMap.put("a", jMediumMsg1)
177 jMap.put("b", jMediumMsg2)
178 jMap.put("c", jMediumMsg3)
179 val jLargeMsg = new JLarge(
```

serialization/jackson/JacksonSerializationBench.scala, line 179 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMediumMsg1
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:179
Taint Flags:

```
176 jMap.put("a", jMediumMsg1)
177 jMap.put("b", jMediumMsg2)
178 jMap.put("c", jMediumMsg3)
179 val jLargeMsg = new JLarge(
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 179 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
jMediumMsg1,
jMediumMsg2,
jMediumMsg3,
```

serialization/jackson/JacksonSerializationBench.scala, line 179 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jMediumMsg1
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:179
Taint Flags:

```
jMap.put("a", jMediumMsg1)
jMap.put("b", jMediumMsg2)
jMap.put("c", jMediumMsg3)
val jLargeMsg = new JLarge(
jMediumMsg1,
jMediumMsg2,
jMediumMsg3,
```

serialization/jackson/JacksonSerializationBench.scala, line 117 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mediumMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:117
Taint Flags:

```
smallMsg1,
smallMsg2,
smallMsg3)
val largeMsg = Large(
mediumMsg1,
mediumMsg2,
mediumMsg3,
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 121 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mediumMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:121
Taint Flags:

```
118  mediumMsg1,
119  mediumMsg2,
120  mediumMsg3,
121  Vector(mediumMsg1, mediumMsg2, mediumMsg3),
122  Map("a" -> mediumMsg1, "b" -> mediumMsg2, "c" -> mediumMsg3))
123
124  val timeMsg = new TimeMessage(5.seconds, LocalDateTime.of(2019, 4, 29, 23, 15, 3, 12345), Instant.now())
```

serialization/jackson/JacksonSerializationBench.scala, line 122 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mediumMsg2
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:122
Taint Flags:

```
119  mediumMsg2,
120  mediumMsg3,
121  Vector(mediumMsg1, mediumMsg2, mediumMsg3),
122  Map("a" -> mediumMsg1, "b" -> mediumMsg2, "c" -> mediumMsg3))
123
124  val timeMsg = new TimeMessage(5.seconds, LocalDateTime.of(2019, 4, 29, 23, 15, 3, 12345), Instant.now())
125
```

serialization/jackson/JacksonSerializationBench.scala, line 130 (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.jackson

serialization/jackson/JacksonSerializationBench.scala, line 130 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: jSmallMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:130
Taint Flags:

```
127  val jSmallMsg1 = new JSmall("abc", 17)
128  val jSmallMsg2 = new JSmall("def", 18)
129  val jSmallMsg3 = new JSmall("ghi", 19)
130  val jMediumMsg1 = new JMedium(
131  "abc",
132  "def",
133  "ghi",
```

serialization/jackson/JacksonSerializationBench.scala, line 145 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jSmallMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:145
Taint Flags:

```
142  jSmallMsg1,
143  jSmallMsg2,
144  jSmallMsg3)
145  val jMediumMsg2 = new JMedium(
146  "ABC",
147  "DEF",
148  "GHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 160 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: jSmallMsg3
Enclosing Method: JacksonSerializationBench()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 160 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: serialization/jackson/JacksonSerializationBench.scala:160
Taint Flags:

```
157 jSmallMsg1,
158 jSmallMsg2,
159 jSmallMsg3)
160 val jMediumMsg3 = new JMedium(
161 "abcABC",
162 "defDEF",
163 "ghiGHI",
```

serialization/jackson/JacksonSerializationBench.scala, line 117 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mediumMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:117
Taint Flags:

```
114 smallMsg1,
115 smallMsg2,
116 smallMsg3)
117 val largeMsg = Large(
118 mediumMsg1,
119 mediumMsg2,
120 mediumMsg3,
```

serialization/jackson/JacksonSerializationBench.scala, line 121 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mediumMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:121
Taint Flags:

```
118 mediumMsg1,
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 121 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
mediumMsg2,
mediumMsg3,
Vector(mediumMsg1, mediumMsg2, mediumMsg3),
Map("a" -> mediumMsg1, "b" -> mediumMsg2, "c" -> mediumMsg3))

val timeMsg = new TimeMessage(5.seconds, LocalDateTime.of(2019, 4, 29, 23, 15, 3, 12345), Instant.now())
```

serialization/jackson/JacksonSerializationBench.scala, line 122 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mediumMsg3
Enclosing Method: JacksonSerializationBench()
File: serialization/jackson/JacksonSerializationBench.scala:122
Taint Flags:

```
mediumMsg2,
mediumMsg3,
Vector(mediumMsg1, mediumMsg2, mediumMsg3),
Map("a" -> mediumMsg1, "b" -> mediumMsg2, "c" -> mediumMsg3))

val timeMsg = new TimeMessage(5.seconds, LocalDateTime.of(2019, 4, 29, 23, 15, 3, 12345), Instant.now())

```

Package: akka.stream

stream/SourceRefBenchmark.scala, line 37 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: SourceRefBenchmark()
File: stream/SourceRefBenchmark.scala:37
Taint Flags:

```
loglevel = "WARNING"
}""".stripMargin).withFallback(ConfigFactory.load())

```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.stream

stream/SourceRefBenchmark.scala, line 37 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
implicit val system: ActorSystem = ActorSystem("test", config)

final val successMarker = Success(1)
final val successFailure = Success(new Exception)
```

stream/FusedGraphsBenchmark.scala, line 83 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: out
Enclosing Method: IdentityStage()
File: stream/FusedGraphsBenchmark.scala:83
Taint Flags:

```
class IdentityStage extends GraphStage[FlowShape[MutableElement, MutableElement]] {
val in = Inlet[MutableElement]("Identity.in")
val out = Outlet[MutableElement]("Identity.out")
override val shape = FlowShape(in, out)

override def createLogic(inheritedAttributes: Attributes): GraphStageLogic =
new GraphStageLogic(shape) with InHandler with OutHandler {
```

stream/PartitionHubBenchmark.scala, line 44 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: PartitionHubBenchmark()
File: stream/PartitionHubBenchmark.scala:44
Taint Flags:

```
}
""")

implicit val system: ActorSystem = ActorSystem("PartitionHubBenchmark", config)

@Param(Array("2", "5", "10", "20", "30"))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.stream

stream/PartitionHubBenchmark.scala, line 44 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
var NumberOfStreams = 0
```

stream/FlowMapBenchmark.scala, line 54 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: FlowMapBenchmark()
File: stream/FlowMapBenchmark.scala:54
Taint Flags:

```
}
}""".stripMargin).withFallback(ConfigFactory.load())

implicit val system: ActorSystem = ActorSystem("test", config)

@Param(Array("true", "false"))
var UseGraphStageIdentity = false
```

stream/MapAsyncBenchmark.scala, line 43 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: MapAsyncBenchmark()
File: stream/MapAsyncBenchmark.scala:43
Taint Flags:

```
}
""")

implicit val system: ActorSystem = ActorSystem("MapAsyncBenchmark", config)
import system.dispatcher

var testSource: Source[java.lang.Integer, NotUsed] = _
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.stream

stream/FlatMapConcatBenchmark.scala, line 43 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: FlatMapConcatBenchmark()
File: stream/FlatMapConcatBenchmark.scala:43
Taint Flags:

```
}
""")

private implicit val system: ActorSystem = ActorSystem("FlatMapConcatBenchmark", config)

var testSource: Source[java.lang.Integer, NotUsed] = _

```

stream/FramingBenchmark.scala, line 55 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config
Enclosing Method: FramingBenchmark()
File: stream/FramingBenchmark.scala:55
Taint Flags:

```
}
}""".stripMargin).withFallback(ConfigFactory.load())

implicit val system: ActorSystem = ActorSystem("test", config)

// Safe to be benchmark scoped because the flows we construct in this bench are stateless
var flow: Source[ByteString, NotUsed] = _
```

stream/AskBenchmark.scala, line 46 (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.stream

stream/AskBenchmark.scala, line 46 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: config
Enclosing Method: AskBenchmark()
File: stream/AskBenchmark.scala:46
Taint Flags:

```
43  }
44  """)
45
46  implicit val system: ActorSystem = ActorSystem("MapAsyncBenchmark", config)
47  import system.dispatcher
48
49  var testSource: Source[java.lang.Integer, NotUsed] = _
```

stream/FusedGraphsBenchmark.scala, line 83 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: in
Enclosing Method: IdentityStage()
File: stream/FusedGraphsBenchmark.scala:83
Taint Flags:

```
80 class IdentityStage extends GraphStage[FlowShape[MutableElement, MutableElement]] {
81  val in = Inlet[MutableElement]("Identity.in")
82  val out = Outlet[MutableElement]("Identity.out")
83  override val shape = FlowShape(in, out)
84
85  override def createLogic(inheritedAttributes: Attributes): GraphStageLogic =
86  new GraphStageLogic(shape) with InHandler with OutHandler {
```

stream/FusedGraphsBenchmark.scala, line 53 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: in
Enclosing Method: JitSafeCompletionLatch()

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.stream

stream/FusedGraphsBenchmark.scala, line 53 (Code Correctness: Constructor Invokes Overridable Function)	Low

File: stream/FusedGraphsBenchmark.scala:53
Taint Flags:

```
50
51 class JitSafeCompletionLatch extends GraphStageWithMaterializedValue[SinkShape[MutableElement], CountDownLatch] {
52  val in = Inlet[MutableElement]("JitSafeCompletionLatch.in")
53  override val shape = SinkShape(in)
54
55  override def createLogicAndMaterializedValue(inheritedAttributes: Attributes): (GraphStageLogic, CountDownLatch) = {
56  val latch = new CountDownLatch(1)
```

stream/FusedGraphsBenchmark.scala, line 34 (Code Correctness: Constructor Invokes Overridable Function)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: out
Enclosing Method: TestSource()
File: stream/FusedGraphsBenchmark.scala:34
Taint Flags:

```
31
32 class TestSource(elems: Array[MutableElement]) extends GraphStage[SourceShape[MutableElement]] {
33  val out = Outlet[MutableElement]("TestSource.out")
34  override val shape = SourceShape(out)
35
36  override def createLogic(inheritedAttributes: Attributes): GraphStageLogic =
37  new GraphStageLogic(shape) with OutHandler {
```

Package: akka.stream.io

stream/io/FileSourcesScaleBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: FILES_NUMBER
Enclosing Method: FileSourcesScaleBenchmark()
File: stream/io/FileSourcesScaleBenchmark.scala:41
Taint Flags:

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.stream.io

stream/io/FileSourcesScaleBenchmark.scala, line 41 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
38 implicit val system: ActorSystem = ActorSystem("file-sources-benchmark")
39
40 val FILES_NUMBER = 40
41 val files: Seq[Path] = {
42 val line = ByteString("x" * 2048 + "\n")
43 (1 to FILES_NUMBER).map(i => {
44 val f = Files.createTempFile(getClass.getName, s"$i.bench.tmp")
```

stream/io/FileSourcesBenchmark.scala, line 42 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: system
Enclosing Method: FileSourcesBenchmark()
File: stream/io/FileSourcesBenchmark.scala:42
Taint Flags:

```
39
40 val f = Files.createTempFile(getClass.getName, ".bench.tmp")
41
42 val ft = Source
43 .fromIterator(() => Iterator.continually(line))
44 .take(10 * 39062) // adjust as needed
45 .runWith(FileIO.toPath(f))
```

Package: akka.util

util/ByteString_take_Benchmark.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: str
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:24
Taint Flags:

```
21 val bss = ByteStrings(Vector.fill(numVec)(ByteString1.fromString(str)))
22
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_take_Benchmark.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
val rand = new Random()
val len = str.size * numVec
val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
val n_neg = rand.nextInt(Int.MaxValue) * -1
val n_avg = len / 2
```

util/ByteString_copyToBuffer_Benchmark.scala, line 26 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bs_mini
Enclosing Method: ByteString_copyToBuffer_Benchmark()
File: util/ByteString_copyToBuffer_Benchmark.scala:26
Taint Flags:

```
val bs_small = ByteString(Array.ofDim[Byte](1024 * 1 * 4))
val bs_large = ByteString(Array.ofDim[Byte](1024 * 4 * 4))

val bss_mini = ByteStrings(Vector.fill(4)(bs_mini.asInstanceOf[ByteString1C].toByteString1), 4 * bs_mini.length)
val bss_small = ByteStrings(Vector.fill(4)(bs_small.asInstanceOf[ByteString1C].toByteString1), 4 * bs_small.length)
val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
val bss_pc_large = bss_large.compact
```

util/ByteString_drop_Benchmark.scala, line 21 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: numVec
Enclosing Method: ByteString_drop_Benchmark()
File: util/ByteString_drop_Benchmark.scala:21
Taint Flags:

```

val str = List.fill[Byte](4)(0).mkString
val numVec = 1024
val bss = ByteStrings(Vector.fill(numVec)(ByteString1.fromString(str)))

```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_drop_Benchmark.scala, line 21 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
23 val rand = new Random()
24 val len = str.size * numVec
```

util/ByteString_drop_Benchmark.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: numVec
Enclosing Method: ByteString_drop_Benchmark()
File: util/ByteString_drop_Benchmark.scala:24
Taint Flags:

```
21 val bss = ByteStrings(Vector.fill(numVec)(ByteString1.fromString(str)))
22
23 val rand = new Random()
24 val len = str.size * numVec
25 val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26 val n_neg = rand.nextInt(Int.MaxValue) * -1
27 val n_avg = len / 2
```

util/ByteString_dropRight_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: rand
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:25
Taint Flags:

```
22
23 val rand = new Random()
24 val len = str.size * numVec
25 val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26 val n_neg = rand.nextInt(Int.MaxValue) * -1
27 val n_avg = len / 2
28 val n_best = 1
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_dropRight_Benchmark.scala, line 26 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: rand
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:26
Taint Flags:

```
23 val rand = new Random()
24 val len = str.size * numVec
25 val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26 val n_neg = rand.nextInt(Int.MaxValue) * -1
27 val n_avg = len / 2
28 val n_best = 1
29 val n_worst = len - 1
```

util/ByteString_take_Benchmark.scala, line 21 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: numVec
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:21
Taint Flags:

```
18
19 val str = List.fill[Byte](4)(0).mkString
20 val numVec = 1024
21 val bss = ByteStrings(Vector.fill(numVec)(ByteString1.fromString(str)))
22
23 val rand = new Random()
24 val len = str.size * numVec
```

util/ByteString_take_Benchmark.scala, line 24 (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

util/ByteString_take_Benchmark.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: numVec
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:24
Taint Flags:

```
21 val bss = ByteStrings(Vector.fill(numVec)(ByteString1.fromString(str)))
22
23 val rand = new Random()
24 val len = str.size * numVec
25 val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26 val n_neg = rand.nextInt(Int.MaxValue) * -1
27 val n_avg = len / 2
```

util/ByteString_copyToBuffer_Benchmark.scala, line 28 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bs_large
Enclosing Method: ByteString_copyToBuffer_Benchmark()
File: util/ByteString_copyToBuffer_Benchmark.scala:28
Taint Flags:

```
25
26 val bss_mini = ByteStrings(Vector.fill(4)(bs_mini.asInstanceOf[ByteString1C].toByteString1), 4 * bs_mini.length)
27 val bss_small = ByteStrings(Vector.fill(4)(bs_small.asInstanceOf[ByteString1C].toByteString1), 4 * bs_small.length)
28 val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
29 val bss_pc_large = bss_large.compact
30
31 val buf = ByteBuffer.allocate(1024 * 4 * 4)
```

util/ByteString_dropSliceTake_Benchmark.scala, line 27 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bs_large
Enclosing Method: ByteString_dropSliceTake_Benchmark()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_dropSliceTake_Benchmark.scala, line 27 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: util/ByteString_dropSliceTake_Benchmark.scala:27
Taint Flags:

```
24
25  val bss_mini = ByteStrings(Vector.fill(4)(bs_mini.asInstanceOf[ByteString1C].toByteString1), 4 * bs_mini.length)
26  val bss_small = ByteStrings(Vector.fill(4)(bs_small.asInstanceOf[ByteString1C].toByteString1), 4 * bs_small.length)
27  val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
28  val bss_pc_large = bss_large.compact
29
30  /*
```

util/ByteString_decode_Benchmark.scala, line 26 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: utf8String
Enclosing Method: ByteString_decode_Benchmark()
File: util/ByteString_decode_Benchmark.scala:26
Taint Flags:

```
23  val bc_large = bss_large.compact // compacted
24
25  val utf8String = "utf-8"
26  val utf8 = Charset.forName(utf8String)
27
28  /*
29  Using Charset helps a bit, but nothing impressive:
```

util/ByteString_take_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: rand
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:25
Taint Flags:

```
22
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_take_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
```

util/ByteString_take_Benchmark.scala, line 26 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: rand
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:26
Taint Flags:

```
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
29  val n_worst = len - 1
```

util/ByteString_dropRight_Benchmark.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: str
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:24
Taint Flags:

```
21  val bss = ByteStrings(Vector.fill(numVec)(ByteString1.fromString(str)))
22
23  val rand = new Random()
24  val len = str.size * numVec
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_dropRight_Benchmark.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
```

util/ByteString_take_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:25
Taint Flags:

```
22
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
```

util/ByteString_take_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:25
Taint Flags:

```
22
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_take_Benchmark.scala, line 27 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:27
Taint Flags:

```
val len = str.size * numVec
val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
val n_neg = rand.nextInt(Int.MaxValue) * -1
val n_avg = len / 2
val n_best = 1
val n_worst = len - 1

```

util/ByteString_take_Benchmark.scala, line 29 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:29
Taint Flags:

```
val n_neg = rand.nextInt(Int.MaxValue) * -1
val n_avg = len / 2
val n_best = 1
val n_worst = len - 1

/*
-------------------------------- BASELINE ------------------------------------------------------------------
```

util/ByteString_dropSliceTake_Benchmark.scala, line 26 (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

util/ByteString_dropSliceTake_Benchmark.scala, line 26 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: bs_small
Enclosing Method: ByteString_dropSliceTake_Benchmark()
File: util/ByteString_dropSliceTake_Benchmark.scala:26
Taint Flags:

```
val bs_large = ByteString(Array.ofDim[Byte](1024 * 4 * 4))

val bss_mini = ByteStrings(Vector.fill(4)(bs_mini.asInstanceOf[ByteString1C].toByteString1), 4 * bs_mini.length)
val bss_small = ByteStrings(Vector.fill(4)(bs_small.asInstanceOf[ByteString1C].toByteString1), 4 * bs_small.length)
val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
val bss_pc_large = bss_large.compact

```

util/ByteString_drop_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_drop_Benchmark()
File: util/ByteString_drop_Benchmark.scala:25
Taint Flags:

```

val rand = new Random()
val len = str.size * numVec
val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
val n_neg = rand.nextInt(Int.MaxValue) * -1
val n_avg = len / 2
val n_best = 1
```

util/ByteString_drop_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_drop_Benchmark()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_drop_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: util/ByteString_drop_Benchmark.scala:25
Taint Flags:

```
22
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
```

util/ByteString_drop_Benchmark.scala, line 27 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_drop_Benchmark()
File: util/ByteString_drop_Benchmark.scala:27
Taint Flags:

```
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
29  val n_worst = len - 1
30
```

util/ByteString_drop_Benchmark.scala, line 29 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_drop_Benchmark()
File: util/ByteString_drop_Benchmark.scala:29
Taint Flags:

```
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_drop_Benchmark.scala, line 29 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
val n_avg = len / 2
val n_best = 1
val n_worst = len - 1

/*
--------------------------------- BASELINE -------------------------------------------------------------------
```

util/ByteString_dropSliceTake_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bs_mini
Enclosing Method: ByteString_dropSliceTake_Benchmark()
File: util/ByteString_dropSliceTake_Benchmark.scala:25
Taint Flags:

```
val bs_small = ByteString(Array.ofDim[Byte](1024 * 1 * 4))
val bs_large = ByteString(Array.ofDim[Byte](1024 * 4 * 4))

val bss_mini = ByteStrings(Vector.fill(4)(bs_mini.asInstanceOf[ByteString1C].toByteString1), 4 * bs_mini.length)
val bss_small = ByteStrings(Vector.fill(4)(bs_small.asInstanceOf[ByteString1C].toByteString1), 4 * bs_small.length)
val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
val bss_pc_large = bss_large.compact
```

util/ByteString_indexOf_Benchmark.scala, line 15 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: start
Enclosing Method: ByteString_indexOf_Benchmark()
File: util/ByteString_indexOf_Benchmark.scala:15
Taint Flags:

```
@Measurement(timeUnit = TimeUnit.MILLISECONDS)
class ByteString_indexOf_Benchmark {
  val start = ByteString("abcdefg") ++ ByteString("hijklmno") ++ ByteString("pqrstuv")
  val bss = start ++ start ++ start ++ start ++ start ++ ByteString("xyz")
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_indexOf_Benchmark.scala, line 15 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
16
17  val bs = bss.compact // compacted
18
```

util/ByteString_indexOf_Benchmark.scala, line 15 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: start
Enclosing Method: ByteString_indexOf_Benchmark()
File: util/ByteString_indexOf_Benchmark.scala:15
Taint Flags:

```
12 @Measurement(timeUnit = TimeUnit.MILLISECONDS)
13 class ByteString_indexOf_Benchmark {
14  val start = ByteString("abcdefg") ++ ByteString("hijklmno") ++ ByteString("pqrstuv")
15  val bss = start ++ start ++ start ++ start ++ start ++ ByteString("xyz")
16
17  val bs = bss.compact // compacted
18
```

util/ByteString_indexOf_Benchmark.scala, line 15 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: start
Enclosing Method: ByteString_indexOf_Benchmark()
File: util/ByteString_indexOf_Benchmark.scala:15
Taint Flags:

```
12 @Measurement(timeUnit = TimeUnit.MILLISECONDS)
13 class ByteString_indexOf_Benchmark {
14  val start = ByteString("abcdefg") ++ ByteString("hijklmno") ++ ByteString("pqrstuv")
15  val bss = start ++ start ++ start ++ start ++ start ++ ByteString("xyz")
16
17  val bs = bss.compact // compacted
18
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_indexOf_Benchmark.scala, line 15 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: start
Enclosing Method: ByteString_indexOf_Benchmark()
File: util/ByteString_indexOf_Benchmark.scala:15
Taint Flags:

```
12 @Measurement(timeUnit = TimeUnit.MILLISECONDS)
13 class ByteString_indexOf_Benchmark {
14   val start = ByteString("abcdefg") ++ ByteString("hijklmno") ++ ByteString("pqrstuv")
15   val bss = start ++ start ++ start ++ start ++ start ++ ByteString("xyz")
16
17   val bs = bss.compact // compacted
18
```

util/ByteString_indexOf_Benchmark.scala, line 15 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: start
Enclosing Method: ByteString_indexOf_Benchmark()
File: util/ByteString_indexOf_Benchmark.scala:15
Taint Flags:

```
12 @Measurement(timeUnit = TimeUnit.MILLISECONDS)
13 class ByteString_indexOf_Benchmark {
14   val start = ByteString("abcdefg") ++ ByteString("hijklmno") ++ ByteString("pqrstuv")
15   val bss = start ++ start ++ start ++ start ++ start ++ ByteString("xyz")
16
17   val bs = bss.compact // compacted
18
```

util/ByteString_indexOf_Benchmark.scala, line 17 (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

util/ByteString_indexOf_Benchmark.scala, line 17 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: bss
Enclosing Method: ByteString_indexOf_Benchmark()
File: util/ByteString_indexOf_Benchmark.scala:17
Taint Flags:

```
14 val start = ByteString("abcdefg") ++ ByteString("hijklmno") ++ ByteString("pqrstuv")
15 val bss = start ++ start ++ start ++ start ++ start ++ ByteString("xyz")
16
17 val bs = bss.compact // compacted
18
19 /*
20 original
```

util/ByteString_copyToBuffer_Benchmark.scala, line 29 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bss_large
Enclosing Method: ByteString_copyToBuffer_Benchmark()
File: util/ByteString_copyToBuffer_Benchmark.scala:29
Taint Flags:

```
26 val bss_mini = ByteStrings(Vector.fill(4)(bs_mini.asInstanceOf[ByteString1C].toByteString1), 4 * bs_mini.length)
27 val bss_small = ByteStrings(Vector.fill(4)(bs_small.asInstanceOf[ByteString1C].toByteString1), 4 * bs_small.length)
28 val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
29 val bss_pc_large = bss_large.compact
30
31 val buf = ByteBuffer.allocate(1024 * 4 * 4)
32
```

util/ByteString_decode_Benchmark.scala, line 23 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bss_large
Enclosing Method: ByteString_decode_Benchmark()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_decode_Benchmark.scala, line 23 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: util/ByteString_decode_Benchmark.scala:23
Taint Flags:

```
20 val bs_large = ByteString(Array.ofDim[Byte](1024 * 4 * 4))
21
22 val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
23 val bc_large = bss_large.compact // compacted
24
25 val utf8String = "utf-8"
26 val utf8 = Charset.forName(utf8String)
```

util/ByteString_dropRight_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:25
Taint Flags:

```
22
23 val rand = new Random()
24 val len = str.size * numVec
25 val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26 val n_neg = rand.nextInt(Int.MaxValue) * -1
27 val n_avg = len / 2
28 val n_best = 1
```

util/ByteString_dropRight_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:25
Taint Flags:

```
22
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_dropRight_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
```

util/ByteString_dropRight_Benchmark.scala, line 27 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:27
Taint Flags:

```
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
29  val n_worst = len - 1
30
```

util/ByteString_dropRight_Benchmark.scala, line 29 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: len
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:29
Taint Flags:

```
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
29  val n_worst = len - 1
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_dropRight_Benchmark.scala, line 29 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
30
31 /*
32 --------------------------------- BASELINE ------------------------------------------------------------------------
```

util/ByteString_drop_Benchmark.scala, line 25 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: rand
Enclosing Method: ByteString_drop_Benchmark()
File: util/ByteString_drop_Benchmark.scala:25
Taint Flags:

```
22
23 val rand = new Random()
24 val len = str.size * numVec
25 val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26 val n_neg = rand.nextInt(Int.MaxValue) * -1
27 val n_avg = len / 2
28 val n_best = 1
```

util/ByteString_drop_Benchmark.scala, line 26 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: rand
Enclosing Method: ByteString_drop_Benchmark()
File: util/ByteString_drop_Benchmark.scala:26
Taint Flags:

```
23 val rand = new Random()
24 val len = str.size * numVec
25 val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26 val n_neg = rand.nextInt(Int.MaxValue) * -1
27 val n_avg = len / 2
28 val n_best = 1
29 val n_worst = len - 1
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_drop_Benchmark.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: str
Enclosing Method: ByteString_drop_Benchmark()
File: util/ByteString_drop_Benchmark.scala:24
Taint Flags:

```
21  val bss = ByteStrings(Vector.fill(numVec)(ByteString1.fromString(str)))
22
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
```

util/ByteString_decode_Benchmark.scala, line 22 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bs_large
Enclosing Method: ByteString_decode_Benchmark()
File: util/ByteString_decode_Benchmark.scala:22
Taint Flags:

```
19
20  val bs_large = ByteString(Array.ofDim[Byte](1024 * 4 * 4))
21
22  val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
23  val bc_large = bss_large.compact // compacted
24
25  val utf8String = "utf-8"
```

util/ByteString_dropRight_Benchmark.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.util

util/ByteString_dropRight_Benchmark.scala, line 21 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: numVec
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:21
Taint Flags:

```
18
19  val str = List.fill[Byte](4)(0).mkString
20  val numVec = 1024
21  val bss = ByteStrings(Vector.fill(numVec)(ByteString1.fromString(str)))
22
23  val rand = new Random()
24  val len = str.size * numVec
```

util/ByteString_dropRight_Benchmark.scala, line 24 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: numVec
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:24
Taint Flags:

```
21  val bss = ByteStrings(Vector.fill(numVec)(ByteString1.fromString(str)))
22
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
```

util/ByteString_dropSliceTake_Benchmark.scala, line 28 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bss_large
Enclosing Method: ByteString_dropSliceTake_Benchmark()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.util

util/ByteString_dropSliceTake_Benchmark.scala, line 28 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: util/ByteString_dropSliceTake_Benchmark.scala:28
Taint Flags:

```
val bss_mini = ByteStrings(Vector.fill(4)(bs_mini.asInstanceOf[ByteString1C].toByteString1), 4 * bs_mini.length)
val bss_small = ByteStrings(Vector.fill(4)(bs_small.asInstanceOf[ByteString1C].toByteString1), 4 * bs_small.length)
val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
val bss_pc_large = bss_large.compact

/*
-------------------------------- BASELINE ------------------------------------------------------------------
```

util/ByteString_copyToBuffer_Benchmark.scala, line 27 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: bs_small
Enclosing Method: ByteString_copyToBuffer_Benchmark()
File: util/ByteString_copyToBuffer_Benchmark.scala:27
Taint Flags:

```
val bs_large = ByteString(Array.ofDim[Byte](1024 * 4 * 4))

val bss_mini = ByteStrings(Vector.fill(4)(bs_mini.asInstanceOf[ByteString1C].toByteString1), 4 * bs_mini.length)
val bss_small = ByteStrings(Vector.fill(4)(bs_small.asInstanceOf[ByteString1C].toByteString1), 4 * bs_small.length)
val bss_large = ByteStrings(Vector.fill(4)(bs_large.asInstanceOf[ByteString1C].toByteString1), 4 * bs_large.length)
val bss_pc_large = bss_large.compact

```

Code Correctness: Erroneous String Compare (8 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");
  }
```

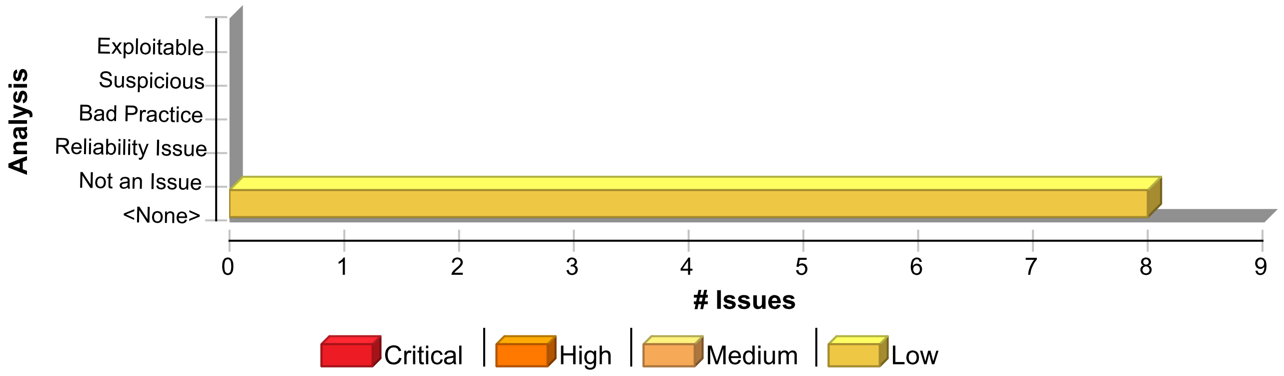
The `==` and `!=` operators will only behave as expected when they are used to compare strings contained in objects that are equal. The most common way for this to occur is for the strings to be interned, whereby the strings are added to a pool of objects maintained by the `String` class. Once a string 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	8	0	0	8
Total	**8**	**0**	**0**	**8**

Code Correctness: Erroneous String Compare — Low

Package: <none>

BenchRunner.scala, line 15 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: apply()
File: BenchRunner.scala:15
Taint Flags:

```
12 def main(args: Array[String]) = {
13 import akka.util.ccompat.JavaConverters._
14
15 val args2 = args.toList.flatMap {
16 case "quick" => "-i 1 -wi 1 -f1 -t1".split(" ").toList
17 case "full" => "-i 10 -wi 4 -f3 -t1".split(" ").toList
18 case "jitwatch" => "-jvmArgs=-XX:+UnlockDiagnosticVMOptions -XX:+TraceClassLoading -XX:+LogCompilation" :: Nil
```

BenchRunner.scala, line 15 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: apply()
File: BenchRunner.scala:15
Taint Flags:

```
12 def main(args: Array[String]) = {
13 import akka.util.ccompat.JavaConverters._
14
15 val args2 = args.toList.flatMap {
16 case "quick" => "-i 1 -wi 1 -f1 -t1".split(" ").toList
17 case "full" => "-i 10 -wi 4 -f3 -t1".split(" ").toList
18 case "jitwatch" => "-jvmArgs=-XX:+UnlockDiagnosticVMOptions -XX:+TraceClassLoading -XX:+LogCompilation" :: Nil
```

BenchRunner.scala, line 15 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Code Correctness: Erroneous String Compare — Low

Package: <none>

BenchRunner.scala, line 15 (Code Correctness: Erroneous String Compare) — Low

Sink: Operation
Enclosing Method: apply()
File: BenchRunner.scala:15
Taint Flags:

```
12 def main(args: Array[String]) = {
13 import akka.util.ccompat.JavaConverters._
14
15 val args2 = args.toList.flatMap {
16 case "quick" => "-i 1 -wi 1 -f1 -t1".split(" ").toList
17 case "full" => "-i 10 -wi 4 -f3 -t1".split(" ").toList
18 case "jitwatch" => "-jvmArgs=-XX:+UnlockDiagnosticVMOptions -XX:+TraceClassLoading -XX:+LogCompilation" :: Nil
```

Package: akka.actor

actor/AffinityPoolRequestResponseBenchmark.scala, line 47 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: setup()
File: actor/AffinityPoolRequestResponseBenchmark.scala:47
Taint Flags:

```
44
45 requireRightNumberOfCores(cores)
46
47 val mailboxConf = mailbox match {
48 case "default" => ""
49 case "SingleConsumerOnlyUnboundedMailbox" =>
50 s"""default-mailbox.mailbox-type = "${classOf[akka.dispatch.SingleConsumerOnlyUnboundedMailbox].getName}""""
```

actor/AffinityPoolComparativeBenchmark.scala, line 43 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: setup()
File: actor/AffinityPoolComparativeBenchmark.scala:43

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

actor/AffinityPoolComparativeBenchmark.scala, line 43 (Code Correctness: Erroneous String Compare) — Low

Taint Flags:

```
40
41 requireRightNumberOfCores(cores)
42
43 val mailboxConf = mailbox match {
44 case "default" => ""
45 case "SingleConsumerOnlyUnboundedMailbox" =>
46 s"""default-mailbox.mailbox-type = "${classOf[akka.dispatch.SingleConsumerOnlyUnboundedMailbox].getName}""""
```

actor/AffinityPoolRequestResponseBenchmark.scala, line 47 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: setup()
File: actor/AffinityPoolRequestResponseBenchmark.scala:47
Taint Flags:

```
44
45 requireRightNumberOfCores(cores)
46
47 val mailboxConf = mailbox match {
48 case "default" => ""
49 case "SingleConsumerOnlyUnboundedMailbox" =>
50 s"""default-mailbox.mailbox-type = "${classOf[akka.dispatch.SingleConsumerOnlyUnboundedMailbox].getName}""""
```

actor/AffinityPoolComparativeBenchmark.scala, line 43 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: setup()
File: actor/AffinityPoolComparativeBenchmark.scala:43
Taint Flags:

```
40
41 requireRightNumberOfCores(cores)
```

Code Correctness: Erroneous String Compare — Low

Package: akka.actor

actor/AffinityPoolComparativeBenchmark.scala, line 43 (Code Correctness: Erroneous String Compare) — Low

```
val mailboxConf = mailbox match {
case "default" => ""
case "SingleConsumerOnlyUnboundedMailbox" =>
s"""default-mailbox.mailbox-type = "${classOf[akka.dispatch.SingleConsumerOnlyUnboundedMailbox].getName}""""
```

Package: akka.remote.artery

remote/artery/CodecBenchmark.scala, line 96 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: setupTrial()
File: remote/artery/CodecBenchmark.scala:96
Taint Flags:

```
actor.serialization-bindings {"${classOf[DummyMessage].getName}" = codec-benchmark }
}
""")
val config = configType match {
case RemoteInstrument =>
ConfigFactory
.parseString(s"""akka.remote.artery.advanced.instruments = [ "${classOf[DummyRemoteInstrument].getName}" ]""")
```

Code Correctness: Misleading Method Signature (1 issue)

Abstract

This looks like an effort to override a common Java method, but it probably does not have the intended effect.

Explanation

This method's name is similar to a common Java method name, but it is either spelled incorrectly or the argument list causes it to not override the intended method. **Example 1:** The following method is meant to override `Object.equals()`:

```
public boolean equals(Object obj1, Object obj2) {
  ...
}
```

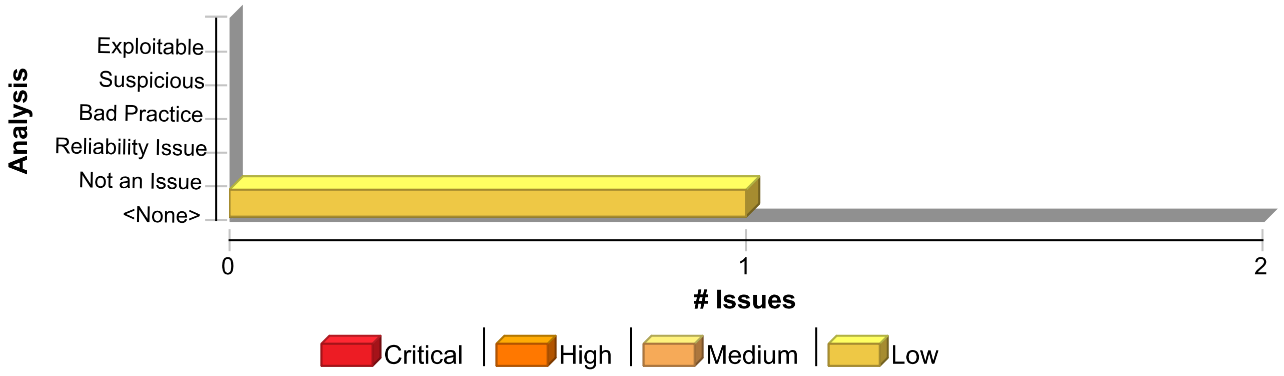
But since `Object.equals()` only takes a single argument, the method in `Example 1` is never called.

Recommendation

Carefully check to make sure the method does what was intended. **Example 2:** The code in `Example 1` could be rewritten in the following way:

```
public boolean equals(Object obj) {
  ...
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: Misleading Method Signature	1	0	0	1
Total	**1**	**0**	**0**	**1**

Code Correctness: Misleading Method Signature **Low**

Package: akka.util

util/ImmutableIntMapBench.scala, line 102 (Code Correctness: Misleading Method Signature) **Low**

Issue Details

Kingdom: Code Quality

Code Correctness: Misleading Method Signature	Low

Package: akka.util

util/ImmutableIntMapBench.scala, line 102 (Code Correctness: Misleading Method Signature)	Low

Scan Engine: SCA (Structural)

Sink Details

Sink: Function: hashcode
Enclosing Method: hashcode()
File: util/ImmutableIntMapBench.scala:102
Taint Flags:

```
99
100  @Benchmark
101  @OperationsPerInvocation(10000)
102  def hashcode(): Int = hashCode(10000, odd1000, 0)
103
104  @Benchmark
105  @OperationsPerInvocation(1000)
```

Code Correctness: Non-Static Inner Class Implements Serializable (7 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

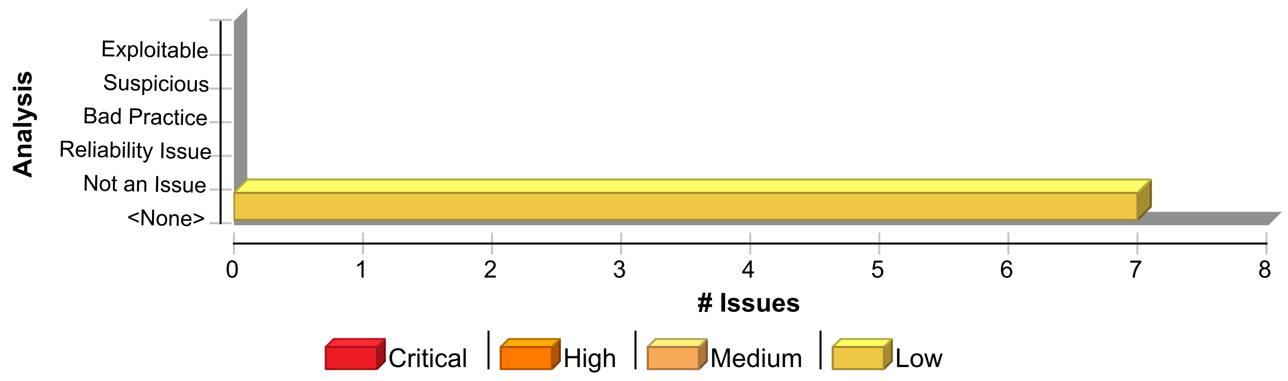
When using inner classes, they should not be serialized, or they should be changed to static-nested classes, since these do not have the drawbacks that non-static inner classes have when serialized. When a nested class is static it inherently has no association with instance variables (including those 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	7	0	0	7
Total	**7**	**0**	**0**	**7**

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

Package: akka.actor

actor/TellOnlyBenchmark.scala, line 118 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TellOnlyBenchmark$UnboundedDroppingMailbox
File: actor/TellOnlyBenchmark.scala:118
Taint Flags:

```
115 }
116 }
117
118 case class UnboundedDroppingMailbox() extends MailboxType with ProducesMessageQueue[DroppingMessageQueue] {
119
120 def this(settings: ActorSystem.Settings, config: Config) = this()
121
```

actor/TellOnlyBenchmark.scala, line 109 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TellOnlyBenchmark$DroppingMessageQueue

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

Package: akka.actor

actor/TellOnlyBenchmark.scala, line 109 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

File: actor/TellOnlyBenchmark.scala:109
Taint Flags:

```
106 }
107 }
108
109 class DroppingMessageQueue extends UnboundedMailbox.MessageQueue {
110 @volatile var dropping = false
111
112 override def enqueue(receiver: ActorRef, handle: Envelope): Unit = {
```

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 28 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: JacksonSerializationBench$Small
File: serialization/jackson/JacksonSerializationBench.scala:28
Taint Flags:

```
25 object JacksonSerializationBench {
26 trait TestMessage
27
28 final case class Small(name: String, num: Int) extends TestMessage
29
30 final case class Medium(
31 field1: String,
```

serialization/jackson/JacksonSerializationBench.scala, line 47 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: JacksonSerializationBench$Large
File: serialization/jackson/JacksonSerializationBench.scala:47
Taint Flags:

```
44 nested3: Small)
```

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

Package: akka.serialization.jackson

serialization/jackson/JacksonSerializationBench.scala, line 47 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
45 extends TestMessage
46
47 final case class Large(
48 nested1: Medium,
49 nested2: Medium,
50 nested3: Medium,
```

serialization/jackson/JacksonSerializationBench.scala, line 30 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: JacksonSerializationBench$Medium
File: serialization/jackson/JacksonSerializationBench.scala:30
Taint Flags:

```
27
28 final case class Small(name: String, num: Int) extends TestMessage
29
30 final case class Medium(
31 field1: String,
32 field2: String,
33 field3: String,
```

Package: akka.stream

stream/InterpreterBenchmark.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: InterpreterBenchmark$GraphDataSink
File: stream/InterpreterBenchmark.scala:75
Taint Flags:

```
72 })
73 }
74
75 case class GraphDataSink[T](override val toString: String, var expected: Int)
76 extends DownstreamBoundaryStageLogic[T] {
```

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

Package: akka.stream

stream/InterpreterBenchmark.scala, line 75 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
77  override val in: akka.stream.Inlet[T] = Inlet[T]("in")
78  in.id = 0
```

stream/InterpreterBenchmark.scala, line 55 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: InterpreterBenchmark$GraphDataSource
File: stream/InterpreterBenchmark.scala:55
Taint Flags:

```
52
53 object InterpreterBenchmark {
54
55  case class GraphDataSource[T](override val toString: String, data: Vector[T]) extends UpstreamBoundaryStageLogic[T] {
56  var idx: Int = 0
57  override val out: akka.stream.Outlet[T] = Outlet[T]("out")
58  out.id = 0
```

Code Correctness: toString on Array (1 issue)

Abstract

`toString()` is called on an array.

Explanation

In most cases, a call to `toString()` on an array indicates a developer is interested in returning the contents of the array as a String. However, a direct call to `toString()` on an array will return a string value containing the array's type and hashcode in memory. **Example 1:** The following code will output `[Ljava.lang.String;@1232121`.

```
String[] strList = new String[5];
...
System.out.println(strList);
```

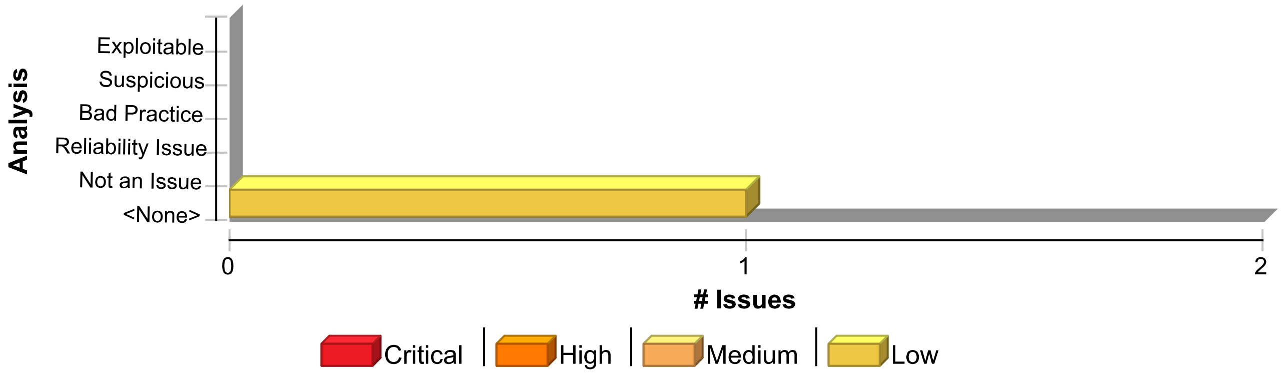
Recommendation

Several `Array.toString()` methods were introduced in Java 5 [1]. These methods will return a string representation of the array contents in comma delimited format. **Example 2:**

```
String[] strList = new String[5];
...
System.out.println(Arrays.toString(strList));
```

If you are using an older version of Java, you will need to iterate through the array to get the value of each element.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Code Correctness: toString on Array	1	0	0	1
Total	**1**	**0**	**0**	**1**

Code Correctness: toString on Array **Low**

Package: akka.remote.artery

remote/artery/CodecBenchmark.scala, line 319 (Code Correctness: toString on Array) **Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Structural)

Code Correctness: toString on Array Low

Package: akka.remote.artery

remote/artery/CodecBenchmark.scala, line 319 (Code Correctness: toString on Array) Low

Sink Details

Sink: FunctionCall: toString
Enclosing Method: remoteReadMetadata()
File: remote/artery/CodecBenchmark.scala:319
Taint Flags:

```
316 else false
317 }
318 if (metaLength != length || !compare(0))
319 throw new IOException(s"DummyInstrument deserialization error. Expected ${Metadata.toString}")
320 }
321
322 override def remoteMessageSent(
```

Dead Code: Expression is Always false (1 issue)

Abstract

This expression will always evaluate to `false`.

Explanation

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

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

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

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

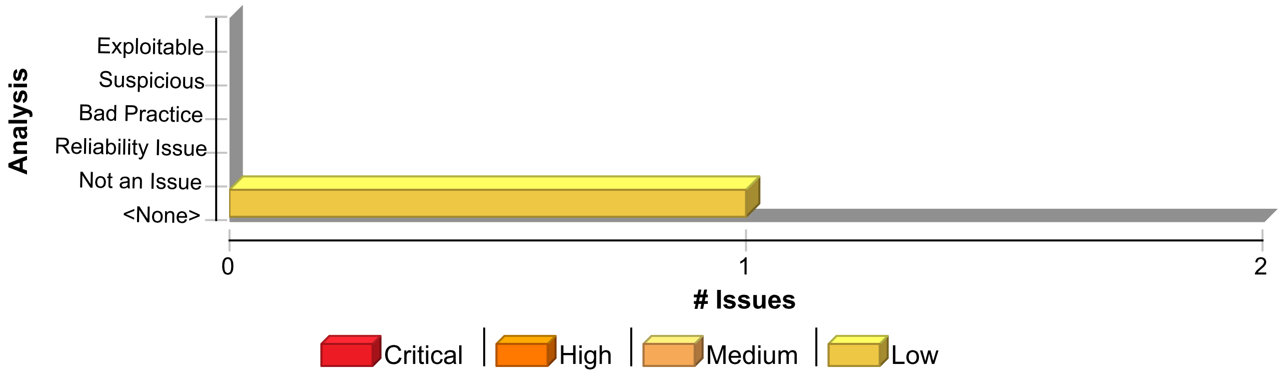
Example 2: The following method never sets the variable `firstCall` to `true`. (The variable `firstCall` is mistakenly set to `false` after the first conditional statement.) The result is that the first part of the expression `firstCall && secondCall` will always evaluate to `false`.

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

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

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

Recommendation

In general, you should repair or remove unused code. It causes additional complexity and maintenance burden without contributing to the functionality of the program.

Issue Summary



Engine Breakdown

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

Dead Code: Expression is Always false — Low

Package: akka.actor

actor/TellOnlyBenchmark.scala, line 103 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: applyOrElse()
File: actor/TellOnlyBenchmark.scala:103
Taint Flags:

```
100
101 class Echo extends Actor {
102 def receive = {
103 case `stop` =>
104 context.stop(self)
105 case m => sender() ! m
106 }
```

Insecure Randomness (21 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.

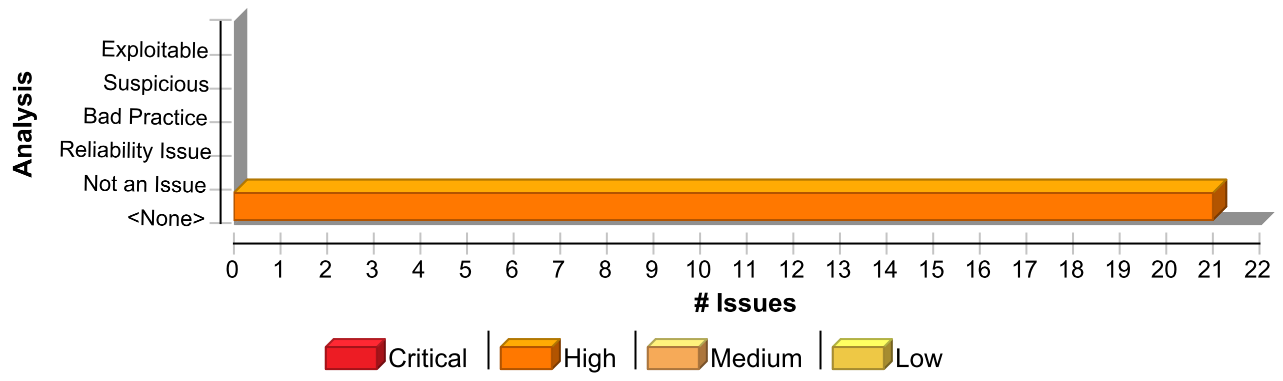
```
def GenerateReceiptURL(baseUrl : String) : String {
    val ranGen = new scala.util.Random()
    ranGen.setSeed((new Date()).getTime())
    return (baseUrl + ranGen.nextInt(400000000) + ".html")
}
```

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	21	0	0	21
Total	**21**	**0**	**0**	**21**

Insecure Randomness — High

Package: actor

actor/RequestResponseActors.scala, line 67 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: apply()
File: actor/RequestResponseActors.scala:67
Taint Flags:

```
64 id <- 0 until numUsersInDB
65 firstName = r.nextString(5)
66 lastName = r.nextString(7)
67 ssn = r.nextInt()
68 friendIds = for { _ <- 0 until 5 } yield r.nextInt(numUsersInDB)
69 } yield id -> User(id, firstName, lastName, ssn, friendIds)
70 Props(new UserServiceActor(users.toMap, latch, numQueries))
```

actor/RequestResponseActors.scala, line 65 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextString()
Enclosing Method: apply()
File: actor/RequestResponseActors.scala:65
Taint Flags:

Insecure Randomness — High

Package: actor

actor/RequestResponseActors.scala, line 65 (Insecure Randomness) — High

```
62 val r = new Random()
63 val users = for {
64 id <- 0 until numUsersInDB
65 firstName = r.nextString(5)
66 lastName = r.nextString(7)
67 ssn = r.nextInt()
68 friendIds = for { _ <- 0 until 5 } yield r.nextInt(numUsersInDB)
```

actor/RequestResponseActors.scala, line 68 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: apply()
File: actor/RequestResponseActors.scala:68
Taint Flags:

```
65 firstName = r.nextString(5)
66 lastName = r.nextString(7)
67 ssn = r.nextInt()
68 friendIds = for { _ <- 0 until 5 } yield r.nextInt(numUsersInDB)
69 } yield id -> User(id, firstName, lastName, ssn, friendIds)
70 Props(new UserServiceActor(users.toMap, latch, numQueries))
71 }
```

actor/RequestResponseActors.scala, line 66 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextString()
Enclosing Method: apply()
File: actor/RequestResponseActors.scala:66
Taint Flags:

```
63 val users = for {
64 id <- 0 until numUsersInDB
65 firstName = r.nextString(5)
66 lastName = r.nextString(7)
67 ssn = r.nextInt()
68 friendIds = for { _ <- 0 until 5 } yield r.nextInt(numUsersInDB)
```

Insecure Randomness — High

Package: actor

actor/RequestResponseActors.scala, line 66 (Insecure Randomness) — High

```
69 } yield id -> User(id, firstName, lastName, ssn, friendIds)
```

Package: akka.actor

actor/RequestResponseActors.scala, line 30 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: applyOrElse()
File: actor/RequestResponseActors.scala:30
Taint Flags:

```
27 latch.countDown()
28 context.stop(self)
29 } else {
30 sender() ! Request(randGenerator.nextInt(numUsersInDB))
31 }
32 left -= 1
33 }
```

actor/DirectByteBufferPoolBenchmark.scala, line 60 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: unpooledHeapAllocAndRelease()
File: actor/DirectByteBufferPoolBenchmark.scala:60
Taint Flags:

```
57
58 @Benchmark
59 def unpooledHeapAllocAndRelease(): Unit = {
60 val idx = random.nextInt(unpooledHeapBuffers.length)
61 val oldBuf = unpooledHeapBuffers(idx)
62 if (oldBuf != null) DirectByteBufferPool.tryCleanDirectByteBuffer(oldBuf)
63 unpooledHeapBuffers(idx) = ByteBuffer.allocateDirect(size)
```

actor/DirectByteBufferPoolBenchmark.scala, line 76 (Insecure Randomness) — High

Issue Details

Insecure Randomness — High

Package: akka.actor

actor/DirectByteBufferPoolBenchmark.scala, line 76 (Insecure Randomness) — High

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: pooledDirectAllocAndRelease()
File: actor/DirectByteBufferPoolBenchmark.scala:76
Taint Flags:

```
73
74 @Benchmark
75 def pooledDirectAllocAndRelease(): Unit = {
76 val idx = random.nextInt(pooledDirectBuffers.length)
77 val oldBuf = pooledDirectBuffers(idx)
78 if (oldBuf != null) arteryPool.release(oldBuf)
79 pooledDirectBuffers(idx) = arteryPool.acquire()
```

actor/DirectByteBufferPoolBenchmark.scala, line 68 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: unpooledDirectAllocAndRelease()
File: actor/DirectByteBufferPoolBenchmark.scala:68
Taint Flags:

```
65
66 @Benchmark
67 def unpooledDirectAllocAndRelease(): Unit = {
68 val idx = random.nextInt(unpooledDirectBuffers.length)
69 val oldBuf = unpooledDirectBuffers(idx)
70 if (oldBuf != null) DirectByteBufferPool.tryCleanDirectByteBuffer(oldBuf)
71 unpooledDirectBuffers(idx) = ByteBuffer.allocateDirect(size)
```

Package: akka.remote.artery.compress

remote/artery/compress/InvertCompressionTableBenchmark.scala, line 21 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Insecure Randomness	High

Package: akka.remote.artery.compress

remote/artery/compress/InvertCompressionTableBenchmark.scala, line 21 (Insecure Randomness)	High

Sink: nextInt()
Enclosing Method: randomName()
File: remote/artery/compress/InvertCompressionTableBenchmark.scala:21
Taint Flags:

```
a.r.artery.compress.CompressionTableBenchmark.invert_comp_to_decomp_256 N/A thrpt 20 29040.889 ± 345.425 ops/s
*/

def randomName = ThreadLocalRandom.current().nextInt(1000).toString
val compTable_256 = CompressionTable(17L, 2, Map(Vector.fill[String](256)(randomName).zipWithIndex: _*))
val compTable_1024 = CompressionTable(17L, 3, Map(Vector.fill[String](1024)(randomName).zipWithIndex: _*))

```

Package: akka.remote.compress

remote/compress/HeavyHittersBenchmark.scala, line 67 (Insecure Randomness)	High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextLong()
Enclosing Method: init()
File: remote/compress/HeavyHittersBenchmark.scala:67
Taint Flags:

```
var i = 0
while (i < n) {
topN.update(i.toString, i)
preallocatedNums(i) = rand.nextLong()
preallocatedStrings(i) = i.toString
i += 1
}
```

Package: akka.util

util/ByteString_drop_Benchmark.scala, line 25 (Insecure Randomness)	High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: ByteString_drop_Benchmark()

Insecure Randomness — High

Package: akka.util

util/ByteString_drop_Benchmark.scala, line 25 (Insecure Randomness) — High

File: util/ByteString_drop_Benchmark.scala:25
Taint Flags:

```
22
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
```

util/ByteString_take_Benchmark.scala, line 25 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:25
Taint Flags:

```
22
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
```

util/LruBoundedCacheBench.scala, line 59 (Insecure Randomness) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextString()
Enclosing Method: setup()
File: util/LruBoundedCacheBench.scala:59
Taint Flags:

```
56  lruCache.get(value)
57  }
58
59  toAdd = random.nextString(stringSize)
```

Insecure Randomness **High**

Package: akka.util

util/LruBoundedCacheBench.scala, line 59 (Insecure Randomness) **High**

```
60
61  }
62
```

util/ByteString_dropRight_Benchmark.scala, line 26 (Insecure Randomness) **High**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:26
Taint Flags:

```
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
29  val n_worst = len - 1
```

util/ByteString_dropRight_Benchmark.scala, line 25 (Insecure Randomness) **High**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: ByteString_dropRight_Benchmark()
File: util/ByteString_dropRight_Benchmark.scala:25
Taint Flags:

```
22
23  val rand = new Random()
24  val len = str.size * numVec
25  val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
26  val n_neg = rand.nextInt(Int.MaxValue) * -1
27  val n_avg = len / 2
28  val n_best = 1
```

util/ByteString_take_Benchmark.scala, line 26 (Insecure Randomness) **High**

Issue Details

Insecure Randomness High

Package: akka.util

util/ByteString_take_Benchmark.scala, line 26 (Insecure Randomness) High

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: ByteString_take_Benchmark()
File: util/ByteString_take_Benchmark.scala:26
Taint Flags:

```
val rand = new Random()
val len = str.size * numVec
val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
val n_neg = rand.nextInt(Int.MaxValue) * -1
val n_avg = len / 2
val n_best = 1
val n_worst = len - 1
```

util/ByteString_drop_Benchmark.scala, line 26 (Insecure Randomness) High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: ByteString_drop_Benchmark()
File: util/ByteString_drop_Benchmark.scala:26
Taint Flags:

```
val rand = new Random()
val len = str.size * numVec
val n_greater_or_eq_to_len = len + rand.nextInt(Int.MaxValue - len)
val n_neg = rand.nextInt(Int.MaxValue) * -1
val n_avg = len / 2
val n_best = 1
val n_worst = len - 1
```

Package: remote.artery.compress

remote/artery/compress/CountMinSketchBenchmark.scala, line 37 (Insecure Randomness) High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Insecure Randomness High

Package: remote.artery.compress

remote/artery/compress/CountMinSketchBenchmark.scala, line 37 (Insecure Randomness) High

Sink: nextInt()
Enclosing Method: apply()
File: remote/artery/compress/CountMinSketchBenchmark.scala:37
Taint Flags:

```
34 countMinSketch = new CountMinSketch(d, w, seed)
35 (0 to 8191).foreach { index =>
36 preallocateIds(index) = rand.nextInt()
37 preallocateValues(index) = Math.abs(rand.nextInt())
38 }
39 }
40
```

remote/artery/compress/CountMinSketchBenchmark.scala, line 36 (Insecure Randomness) High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextInt()
Enclosing Method: apply()
File: remote/artery/compress/CountMinSketchBenchmark.scala:36
Taint Flags:

```
33 def init(): Unit = {
34 countMinSketch = new CountMinSketch(d, w, seed)
35 (0 to 8191).foreach { index =>
36 preallocateIds(index) = rand.nextInt()
37 preallocateValues(index) = Math.abs(rand.nextInt())
38 }
39 }
```

Package: stream

stream/FramingBenchmark.scala, line 70 (Insecure Randomness) High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextPrintableChar()
Enclosing Method: apply()
File: stream/FramingBenchmark.scala:70
Taint Flags:

```
67 def setup(): Unit = {
```

Insecure Randomness High

Package: stream

stream/FramingBenchmark.scala, line 70 (Insecure Randomness) High

```
SystemMaterializer(system).materializer

val frame = List.range(0, messageSize, 1).map(_ => Random.nextPrintableChar()).mkString + "\n"
val messageChunk = ByteString(List.range(0, framePerSeq, 1).map(_ => frame).mkString)

Source
```

Package: util

util/LruBoundedCacheBench.scala, line 52 (Insecure Randomness) High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: nextString()
Enclosing Method: apply()
File: util/LruBoundedCacheBench.scala:52
Taint Flags:

```

// Loading
for (i <- 1 to threshold) {
val value = random.nextString(stringSize)
if (i == 1) toGet = value
toRemove = value
javaHashMap.put(value, value)
```

Insecure Randomness: Hardcoded Seed (1 issue)

Abstract

Functions that generate random or pseudorandom values, which are passed a seed, should not be called with a constant argument.

Explanation

Functions that generate random or pseudorandom values, which are passed a seed, should not be called with a constant argument. If a pseudorandom number generator (such as `Random`) is seeded with a specific value (using a function such as `Random.setSeed()`), the values returned by `Random.nextInt()` and similar methods which return or assign values are predictable for an attacker that can collect a number of PRNG outputs. **Example 1:** The values produced by the `Random` object `s` are predictable from the `Random` object `r`.

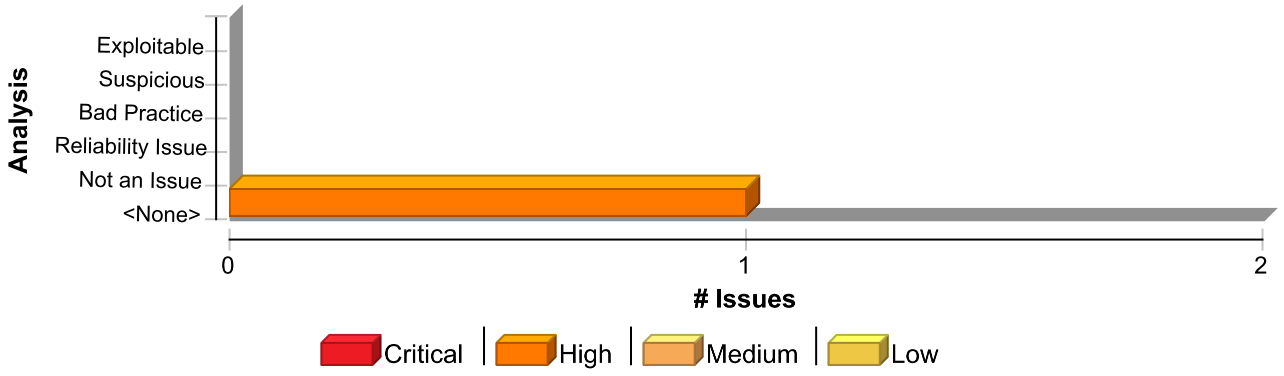
```
Random r = new Random();
r.setSeed(12345);
int i = r.nextInt();
byte[] b = new byte[4];
r.nextBytes(b);

Random s = new Random();
s.setSeed(12345);
int j = s.nextInt();
byte[] c = new byte[4];
s.nextBytes(c);
```

In this example, pseudorandom number generators: `r` and `s` were identically seeded, so `i == j`, and corresponding values of arrays `b[]` and `c[]` are equal.

Recommendation

Use a cryptographic PRNG seeded with hardware-based sources of randomness, such as ring oscillators, disk drive timing, thermal noise, or radioactive decay. Doing so makes the sequence of data produced by `Random.nextInt()` and similar methods much harder to predict than setting the seed to a constant.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Insecure Randomness: Hardcoded Seed	1	0	0	1
Total	**1**	**0**	**0**	**1**

Insecure Randomness: Hardcoded Seed — High

Package: akka.remote.compress

remote/compress/HeavyHittersBenchmark.scala, line 56 (Insecure Randomness: Hardcoded Seed) — **High**

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Semantic)

Sink Details

Sink: Random()
Enclosing Method: HeavyHittersBenchmark()
File: remote/compress/HeavyHittersBenchmark.scala:56
Taint Flags:

```
53
54  private var topN: TopHeavyHitters[String] = _
55
56  val rand = new Random(1001021)
57
58  val preallocatedNums: Array[Long] = Array.ofDim(8192)
59  val preallocatedStrings: Array[String] = Array.ofDim(8192)
```

J2EE Bad Practices: Leftover Debug Code (1 issue)

Abstract

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

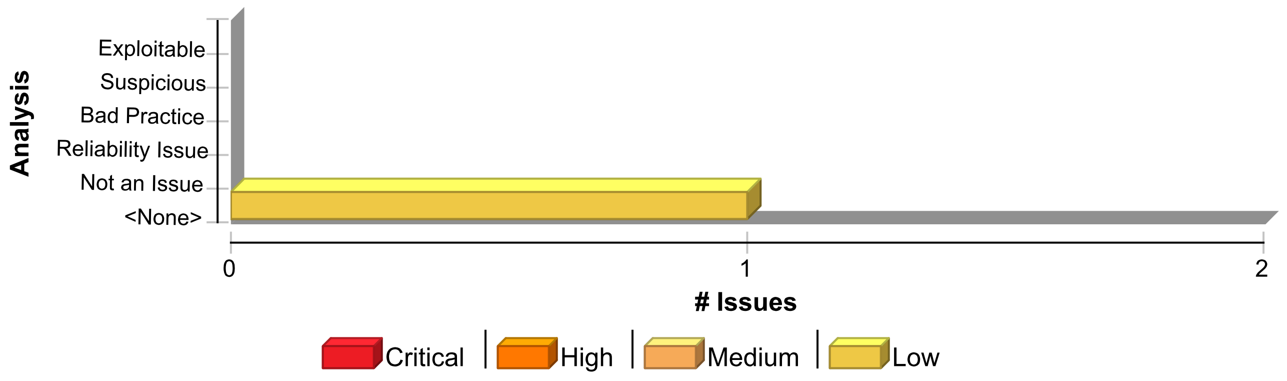
Explanation

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

BenchRunner.scala, line 12 (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

BenchRunner.scala, line 12 (J2EE Bad Practices: Leftover Debug Code) — Low

Sink: Function: main
Enclosing Method: main()
File: BenchRunner.scala:12
Taint Flags:

```
9 import org.openjdk.jmh.runner.options.CommandLineOptions
10
11 object BenchRunner {
12  def main(args: Array[String]) = {
13  import akka.util.ccompat.JavaConverters._
14
15  val args2 = args.toList.flatMap {
```

J2EE Bad Practices: Threads (5 issues)

Abstract

Thread management in a web application is forbidden in some circumstances and is always highly error prone.

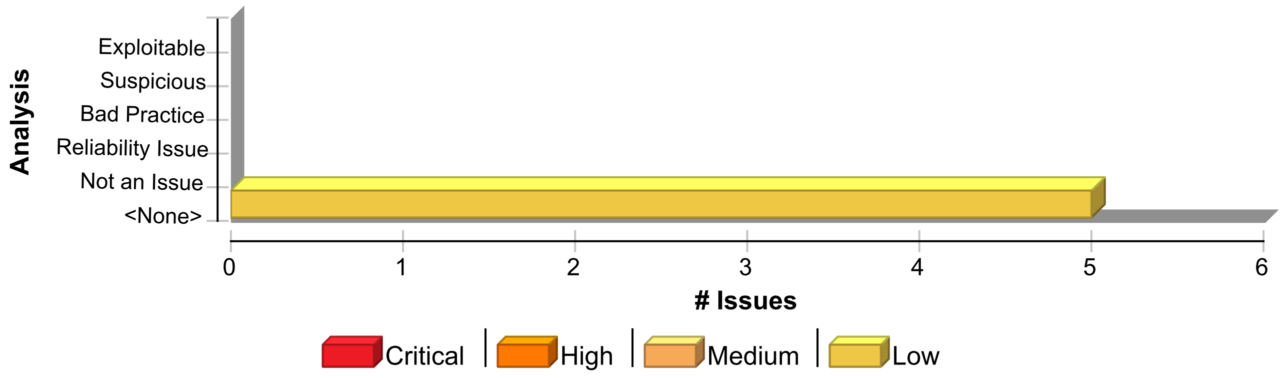
Explanation

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	5	0	0	5
Total	**5**	**0**	**0**	**5**

J2EE Bad Practices: Threads	**Low**
Package: actor	
actor/BenchmarkActors.scala, line 144 (J2EE Bad Practices: Threads)	**Low**

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: availableProcessors()
Enclosing Method: apply()
File: actor/BenchmarkActors.scala:144
Taint Flags:

J2EE Bad Practices: Threads — Low

Package: actor

actor/BenchmarkActors.scala, line 144 (J2EE Bad Practices: Threads) — Low

```
141 def requireRightNumberOfCores(numCores: Int): Unit =
142 require(
143 Runtime.getRuntime.availableProcessors == numCores,
144 s"Update the cores constant to ${Runtime.getRuntime.availableProcessors}")
145
146 def benchmarkPingPongActors(
147 numMessagesPerActorPair: Int,
```

Package: akka.actor

actor/BenchmarkActors.scala, line 142 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: availableProcessors()
Enclosing Method: requireRightNumberOfCores()
File: actor/BenchmarkActors.scala:142
Taint Flags:

```
139 }
140
141 def requireRightNumberOfCores(numCores: Int): Unit =
142 require(
143 Runtime.getRuntime.availableProcessors == numCores,
144 s"Update the cores constant to ${Runtime.getRuntime.availableProcessors}")
145
```

Package: akka.persistence

persistence/LevelDbBatchingBenchmark.scala, line 64 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: sleep()
Enclosing Method: tearDown()
File: persistence/LevelDbBatchingBenchmark.scala:64
Taint Flags:

```
61 @TearDown(Level.Trial)
62 def tearDown(): Unit = {
63 store ! PoisonPill
```

J2EE Bad Practices: Threads — Low

Package: akka.persistence

persistence/LevelDbBatchingBenchmark.scala, line 64 (J2EE Bad Practices: Threads) — Low

```
Thread.sleep(500)

sys.terminate()
Await.ready(sys.whenTerminated, 10.seconds)
```

Package: akka.stream.impl

stream/impl/OutputStreamSourceStageBenchmark.scala, line 45 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: start()
Enclosing Method: consumeWrites()
File: stream/impl/OutputStreamSourceStageBenchmark.scala:45
Taint Flags:

```
}
os.close()
}
}).start()
Await.result(done, 30.seconds)
}

```

stream/impl/OutputStreamSourceStageBenchmark.scala, line 36 (J2EE Bad Practices: Threads) — Low

Issue Details

Kingdom: Time and State
Scan Engine: SCA (Semantic)

Sink Details

Sink: run()
Enclosing Method: consumeWrites()
File: stream/impl/OutputStreamSourceStageBenchmark.scala:36
Taint Flags:

```
@OperationsPerInvocation(WritesPerBench)
def consumeWrites(): Unit = {
val (os, done) = StreamConverters.asOutputStream().toMat(Sink.ignore)(Keep.both).run()
new Thread(new Runnable {
def run(): Unit = {
var counter = 0
```

J2EE Bad Practices: Threads	Low

Package: akka.stream.impl

stream/impl/OutputStreamSourceStageBenchmark.scala, line 36 (J2EE Bad Practices: Threads)	Low

```
39  while (counter > WritesPerBench) {
```

Key Management: Hardcoded Encryption Key (1 issue)

Abstract

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

Explanation

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

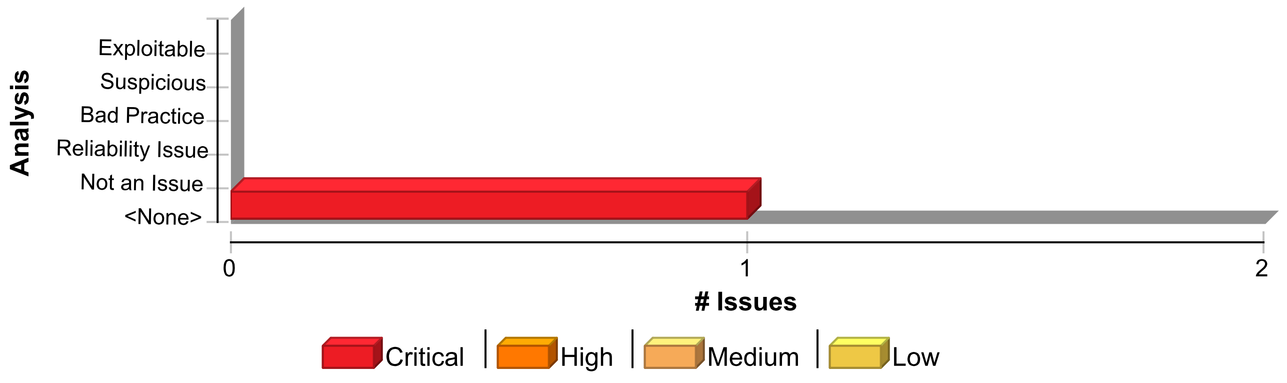
```
...
private static final String encryptionKey = "lakdsljkalkjlksdfkl";
byte[] keyBytes = encryptionKey.getBytes();
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher encryptCipher = Cipher.getInstance("AES");
encryptCipher.init(Cipher.ENCRYPT_MODE, key);
...
```

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

Recommendation

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

Issue Summary



Engine Breakdown

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

Key Management: Hardcoded Encryption Key — Critical

Package: akka.util

util/ImmutableIntMapBench.scala, line 48 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: getKey()
File: util/ImmutableIntMapBench.scala:48
Taint Flags:

```
45 else in
46
47 @tailrec private[this] final def getKey(iterations: Int, key: Int, from: ImmutableIntMap): ImmutableIntMap = {
48 if (iterations > 0 && key != Int.MinValue) {
49 val k = from.get(key)
50 getKey(iterations - 1, k, from)
51 } else from
```

Poor Style: Value Never Read (5 issues)

Abstract

The variable's value is assigned but never used, making it a dead store.

Explanation

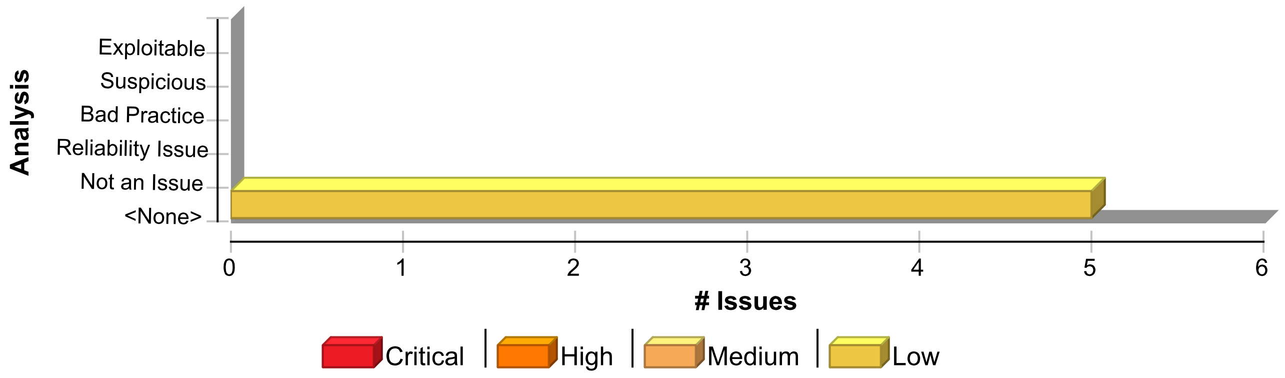
This variable's value is not used. After the assignment, the variable is either assigned another value or goes out of scope. **Example:** The following code excerpt assigns to the variable `r` and then overwrites the value without using it.

```
  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	5	0	0	5
Total	**5**	**0**	**0**	**5**

Poor Style: Value Never Read **Low**

Package: akka.util

util/ByteString_drop_Benchmark.scala, line 63 (Poor Style: Value Never Read) **Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: m
Enclosing Method: bss_best()
File: util/ByteString_drop_Benchmark.scala:63
Taint Flags:

Poor Style: Value Never Read — Low

Package: akka.util

util/ByteString_drop_Benchmark.scala, line 63 (Poor Style: Value Never Read) — Low

```
@Benchmark
def bss_best(): Unit = {
@volatile var m: ByteString = null
m = bss.drop(n_best)
}

@Benchmark
```

util/ByteString_drop_Benchmark.scala, line 45 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: m
Enclosing Method: bss_negative()
File: util/ByteString_drop_Benchmark.scala:45
Taint Flags:

```
@Benchmark
def bss_negative(): Unit = {
@volatile var m: ByteString = null
m = bss.drop(n_neg)
}

@Benchmark
```

util/ByteString_drop_Benchmark.scala, line 57 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: m
Enclosing Method: bss_avg()
File: util/ByteString_drop_Benchmark.scala:57
Taint Flags:

```
@Benchmark
def bss_avg(): Unit = {
@volatile var m: ByteString = null
m = bss.drop(n_avg)
}

```

Poor Style: Value Never Read — Low

Package: akka.util

util/ByteString_drop_Benchmark.scala, line 57 (Poor Style: Value Never Read) — Low

```
60 @Benchmark
```

util/ByteString_drop_Benchmark.scala, line 51 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: m
Enclosing Method: bss_greater_or_eq_to_len()
File: util/ByteString_drop_Benchmark.scala:51
Taint Flags:

```
48 @Benchmark
49 def bss_greater_or_eq_to_len(): Unit = {
50 @volatile var m: ByteString = null
51 m = bss.drop(n_greater_or_eq_to_len)
52 }
53
54 @Benchmark
```

util/ByteString_drop_Benchmark.scala, line 69 (Poor Style: Value Never Read) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: m
Enclosing Method: bss_worst()
File: util/ByteString_drop_Benchmark.scala:69
Taint Flags:

```
66 @Benchmark
67 def bss_worst(): Unit = {
68 @volatile var m: ByteString = null
69 m = bss.drop(n_worst)
70 }
71 }
72
```

System Information Leak (1 issue)

Abstract

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

Explanation

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

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

Depending upon the system configuration, this information can be dumped to a console, written to a log file, or exposed to a remote user. For example, with scripting mechanisms it is trivial to redirect output information from "Standard error" or "Standard output" into a file or another program. Alternatively, the system that the program runs on could have a remote logging mechanism such as a "syslog" server that sends the logs to a remote device. During development, you have no way of knowing where this information might end up being displayed. In some cases, the error message provides the attacker with the precise type of attack to which the system is vulnerable. For example, a database error message can reveal that the application is vulnerable to a SQL injection attack. Other error messages can reveal more oblique clues about the system. In `Example 1`, the leaked information could imply information about the type of operating system, the applications installed on the system, and the amount of care that the administrators have put into configuring the program. Here is another scenario, specific to the mobile world. Most mobile devices now implement a Near-Field Communication (NFC) protocol for quickly sharing information between devices using radio communication. It works by bringing devices to close proximity or simply having them touch each other. Even though the communication range of NFC is limited to just a few centimeters, eavesdropping, data modification and various other types of attacks are possible, since NFC alone does not ensure secure communication. **Example 2:** The Android platform provides support for NFC. The following code creates a message that gets pushed to the other device within the range.

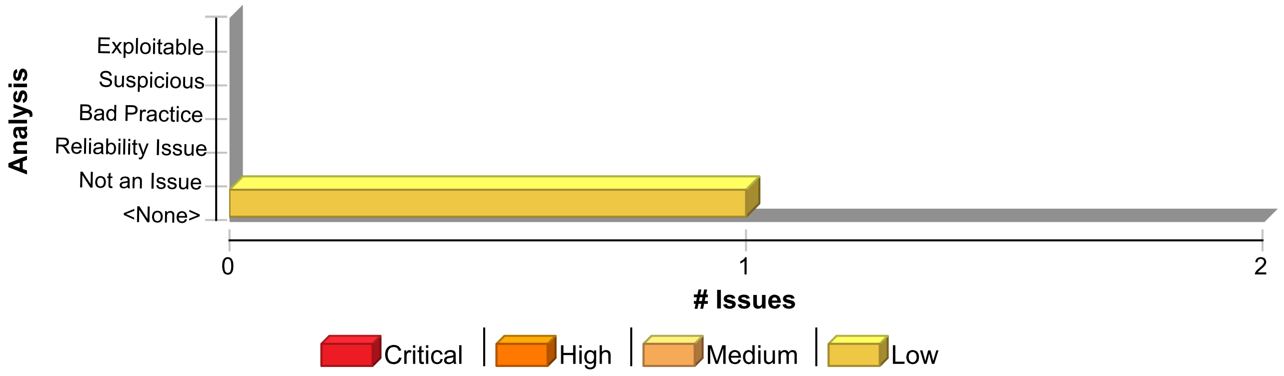
```
...
public static final String TAG = "NfcActivity";
private static final String DATA_SPLITTER = "__:DATA:__";
private static final String MIME_TYPE = "application/my.applications.mimetype";
...
public NdefMessage createNdefMessage(NfcEvent event) {
    TelephonyManager tm =
(TelephonyManager)Context.getSystemService(Context.TELEPHONY_SERVICE);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

NFC Data Exchange Format (NDEF) message contains typed data, a URI, or a custom application payload. If the message contains information about the application, such as its name, MIME type, or device software version, this information could be leaked to an eavesdropper. In `Example 2`, Fortify Static Code Analyzer reports a System Information Leak vulnerability on the return statement.

Recommendation

Write error messages with security in mind. In production environments, turn off detailed error information in favor of brief messages. Restrict the generation and storage of detailed output that can help administrators and programmers diagnose problems. 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. If you are concerned about leaking system data via NFC on an Android device, you could do one of the following three things. Do not include system data in the messages pushed to other devices in range, encrypt the payload of the message, or establish a secure communication channel at a higher layer.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
System Information Leak	1	0	0	1
Total	**1**	**0**	**0**	**1**

System Information Leak — Low

Package: akka.remote.artery

remote/artery/LatchSink.scala, line 30 (System Information Leak) — **Low**

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Semantic)

Sink Details

Sink: printStackTrace()
Enclosing Method: onUpstreamFailure()
File: remote/artery/LatchSink.scala:30
Taint Flags:

```
27
28  override def onUpstreamFailure(ex: Throwable): Unit = {
29  println(ex.getMessage)
30  ex.printStackTrace()
31  }
32
33  override def onPush(): Unit = {
```

System Information Leak: Internal (1 issue)

Abstract

Revealing system data or debugging information could enable an adversary to use system information to plan an 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 prints System information to the standard output stream:

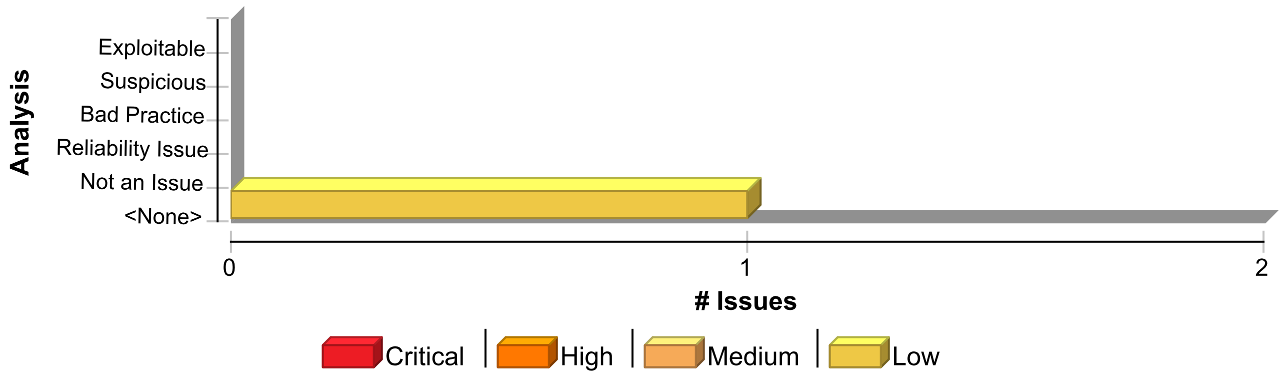
```
...
println(Properties.osName)
...
```

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.

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. 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: Internal	1	0	0	1
Total	**1**	**0**	**0**	**1**

System Information Leak: Internal — Low

Package: akka.remote.artery

remote/artery/LatchSink.scala, line 29 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.Throwable.getMessage()
From: akka.remote.artery.LatchSink$$anon$1.onUpstreamFailure
File: remote/artery/LatchSink.scala:29

```
26  override def preStart(): Unit = pull(in)
27
28  override def onUpstreamFailure(ex: Throwable): Unit = {
29  println(ex.getMessage)
30  ex.printStackTrace()
31  }
32
```

Sink Details

Sink: scala.Predef.println()
Enclosing Method: onUpstreamFailure()
File: remote/artery/LatchSink.scala:29
Taint Flags: EXCEPTIONINFO, SYSTEMINFO

```
26  override def preStart(): Unit = pull(in)
27
28  override def onUpstreamFailure(ex: Throwable): Unit = {
29  println(ex.getMessage)
30  ex.printStackTrace()
31  }
32
```

Unreleased Resource: Streams (1 issue)

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.

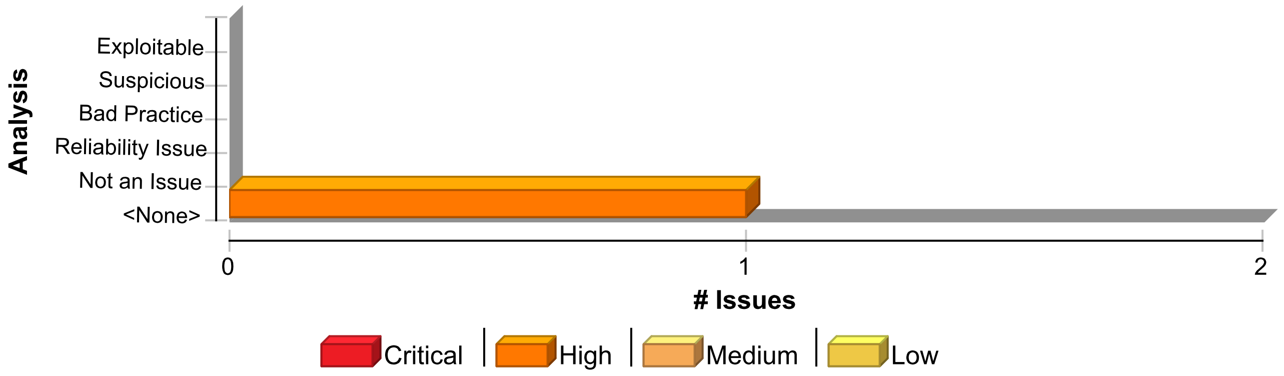
```
def readFile(filename: String): Unit = {
    val data = Source.fromFile(fileName).getLines.mkString
    // Use the data
}
```

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:

```
def readFile(filename: String): Unit = {
    val source = Source.fromFile(filename)
    val data= try source.getLines.mkString finally source.close()
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Unreleased Resource: Streams	1	0	0	1
Total	**1**	**0**	**0**	**1**

Unreleased Resource: Streams — High

Package: stream.io

stream/io/FileSourcesBenchmark.scala, line 63 (Unreleased Resource: Streams) — High

Issue Details

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

Sink Details

Sink: fromFile(...)
Enclosing Method: apply()
File: stream/io/FileSourcesBenchmark.scala:63
Taint Flags:

```
60 fileChannelSource = FileIO.fromPath(file, bufSize)
61 fileInputStreamSource = StreamConverters.fromInputStream(() => Files.newInputStream(file), bufSize)
62 ioSourceLinesIterator =
63 Source.fromIterator(() => scala.io.Source.fromFile(file.toFile).getLines()).map(ByteString(_))
64 }
65
66 @TearDown
```

Weak SecurityManager Check: Overridable Method (3 issues)

Abstract

Non-final methods that perform security checks may be overridden in ways that bypass security checks.

Explanation

If a method is overridden by a child class, the child class can bypass security checks in the parent class. **Example 1:** In the following code, `doSecurityCheck()` performs a security check and can be overridden by a child class.

```
public class BadSecurityCheck {
    private int id;

    public BadSecurityCheck() {
        doSecurityCheck();
        id = 1;
    }
    protected void doSecurityCheck() {
        SecurityManager sm = System.getSecurityManager();
        if (sm != null) {
            sm.checkPermission(new SomePermission("SomeAction"));
        }
    }
}
```

In this example, if the `SecurityManager` permission is not allowed, a `SecurityException` exception will be thrown, which is a runtime exception and will stop the program from executing any further. Since `BadSecurityCheck` is not `final`, and the method `doSecurityCheck()` is `protected` and not `final`, it means that this class can be subclassed to override this function. **Example 2:** In the following code, `doSecurityCheck()` is overridden by a subclass:

```
public class EvilSubclass extends BadSecurityCheck {
    private int id;

    public EvilSubclass() {
        super();
    }
    protected void doSecurityCheck() {
        //do nothing
    }
}
```

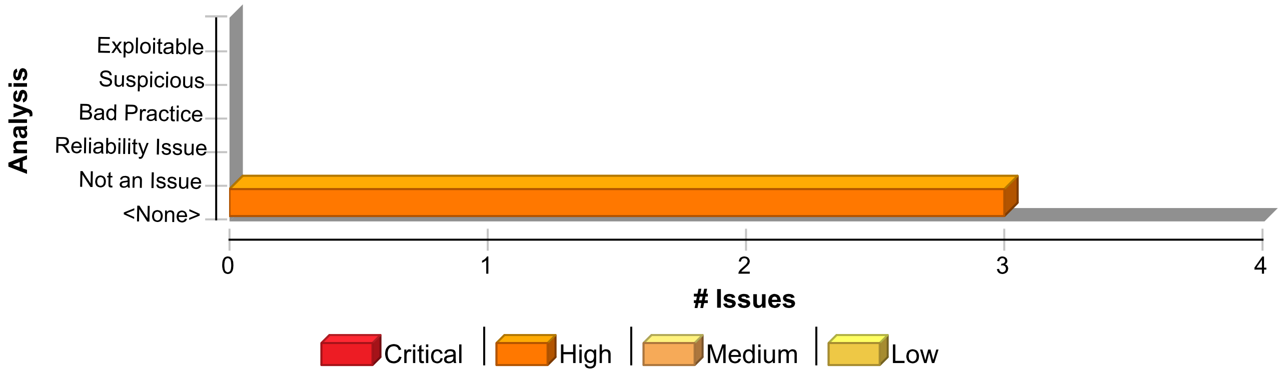
When `EvilSubclass` is instantiated, the constructor first calls `super()`, to invoke the constructor of the superclass. This in turn calls the function `doSecurityCheck()`, but Java will first look for the function within the subclass prior to looking in the superclass, thus invoking the attacker controlled method that bypasses the security check, so `id` will still be set to `1`. This category was derived from the Cigital Java Rulepack.

Recommendation

Make sure any methods that perform security operations (e.g. methods from SecurityManager or AccessController) are declared in `final` classes or the methods themselves are declared final. **Example 2:** The following code declared the class `GoodSecurityCheck` as `final` so none of its methods can be overridden.

```
public final class GoodSecurityCheck {
    private int id;

    public GoodSecurityCheck() {
        doSecurityCheck();
        id = 1;
    }
    void doSecurityCheck() {
        SecurityManager sm = System.getSecurityManager();
        if (sm != null) {
            sm.checkPermission(new SomePermission("SomeAction"));
        }
    }
}
```

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Weak SecurityManager Check: Overridable Method	3	0	0	3
Total	**3**	**0**	**0**	**3**

Weak SecurityManager Check: Overridable Method — High

Package: akka.util

util/StackBench.scala, line 26 (Weak SecurityManager Check: Overridable Method) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: securityManager
Enclosing Method: securityManager()
File: util/StackBench.scala:26
Taint Flags:

```
23 }
24
25 @Benchmark
26 def securityManager(): Array[Class[_]] = {
27 (new CustomSecurtyManager).getTrace
28 }
29
```

Weak SecurityManager Check: Overridable Method — High

Package: akka.util

util/StackBench.scala, line 15 (Weak SecurityManager Check: Overridable Method) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: StackBench$CustomSecurtyManager
Enclosing Method: StackBench$CustomSecurtyManager()
File: util/StackBench.scala:15
Taint Flags:

```
12 @Measurement(timeUnit = TimeUnit.MICROSECONDS)
13 class StackBench {
14
15  class CustomSecurtyManager extends SecurityManager {
16  def getTrace: Array[Class[_]] =
17  getClassContext
18  }
```

util/StackBench.scala, line 16 (Weak SecurityManager Check: Overridable Method) — High

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: Function: getTrace
Enclosing Method: getTrace()
File: util/StackBench.scala:16
Taint Flags:

```
13 class StackBench {
14
15  class CustomSecurtyManager extends SecurityManager {
16  def getTrace: Array[Class[_]] =
17  getClassContext
18  }
19
```