Fortify Standalone Report Generator

Developer Workbook

akka-multi-node-testkit

Table of Contents

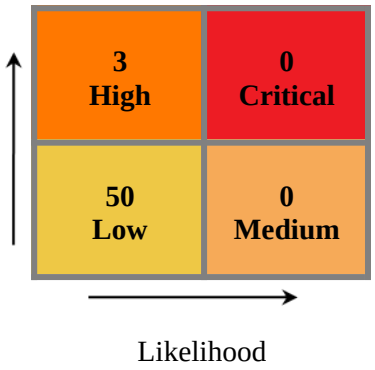
Executive Summary

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

Issues by Priority



Top Ten Critical Categories

This project does not contain any critical issues

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, 11:30 AM	**Engine Version:**	21.1.1.0009
Host Name:	Jacks-Work-MBP.local	**Certification:**	VALID
Number of Files:	7	**Lines of Code:**	825

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

Issue Breakdown by Fortify Categories

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

Category	Fortify Priority (audited/total)				Total Issues
	Critical	High	Medium	Low	
Code Correctness: Constructor Invokes Overridable Function	0	0	0	0 / 29	0 / 29
Code Correctness: Erroneous String Compare	0	0	0	0 / 1	0 / 1
Code Correctness: Non-Static Inner Class Implements Serializable	0	0	0	0 / 16	0 / 16
Dead Code: Expression is Always false	0	0	0	0 / 1	0 / 1
J2EE Bad Practices: JVM Termination	0	0	0	0 / 1	0 / 1
J2EE Bad Practices: Sockets	0	0	0	0 / 1	0 / 1
Often Misused: Authentication	0	0 / 3	0	0	0 / 3
System Information Leak: Internal	0	0	0	0 / 1	0 / 1

Results Outline

Code Correctness: Constructor Invokes Overridable Function (29 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");
    }
  }
}
```

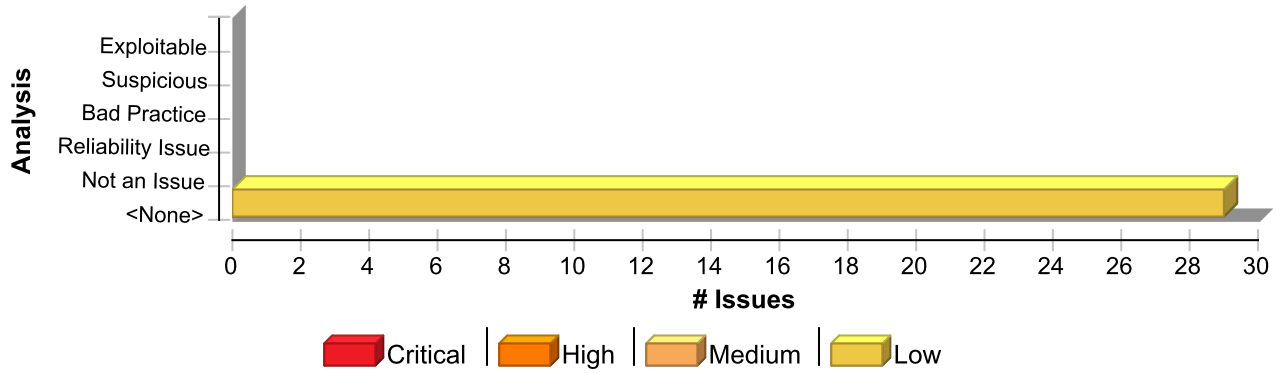
The code in `Example 1` prints "Attack successful!", since the `Attacker` class overrides the `validateUser()` function that is called from the constructor of the superclass `User`, and Java will first look in the subclass for functions called from the constructor.

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	29	0	0	29
Total	**29**	**0**	**0**	**29**

Code Correctness: Constructor Invokes Overridable Function	**Low**
Package: akka.remote.testconductor	
testconductor/Player.scala, line 180 (Code Correctness: Constructor Invokes Overridable Function)	**Low**
Issue Details	

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.testconductor

testconductor/Player.scala, line 180 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ClientFSM()
File: testconductor/Player.scala:180
Taint Flags:

```
177
178 val settings = TestConductor().Settings
179
180 val handler = new PlayerHandler(
181 controllerAddr,
182 settings.ClientReconnects,
183 settings.ReconnectBackoff,
```

testconductor/Player.scala, line 191 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ClientFSM()
File: testconductor/Player.scala:191
Taint Flags:

```
188
189 startWith(Connecting, Data(None, None))
190
191 when(Connecting, stateTimeout = settings.ConnectTimeout) {
192 case Event(_: ClientOp, _) =>
193 stay().replying(Status.Failure(new IllegalStateException("not connected yet")))
194 case Event(Connected(channel), _) =>
```

testconductor/Player.scala, line 205 (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.remote.testconductor

testconductor/Player.scala, line 205 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: settings
Enclosing Method: ClientFSM()
File: testconductor/Player.scala:205
Taint Flags:

```
202 goto(Failed)
203 }
204
205 when(AwaitDone, stateTimeout = settings.BarrierTimeout.duration) {
206 case Event(Done, _) =>
207 log.debug("received Done: starting test")
208 goto(Connected)
```

testconductor/Player.scala, line 336 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: reconnect
Enclosing Method: PlayerHandler()
File: testconductor/Player.scala:336
Taint Flags:

```
333
334 import ClientFSM._
335
336 reconnect()
337
338 var nextAttempt: Deadline = _
339
```

testconductor/Extension.scala, line 89 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: transport
Enclosing Method: TestConductorExt()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.testconductor

testconductor/Extension.scala, line 89 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: testconductor/Extension.scala:89
Taint Flags:

```
86 /**
87 * Transport address of this Netty-like remote transport.
88 */
89 val address = transport.defaultAddress
90
91 }
92
```

testconductor/Conductor.scala, line 429 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: Controller()
File: testconductor/Conductor.scala:429
Taint Flags:

```
426 import Controller._
427
428 val settings = TestConductor().Settings
429 val connection = RemoteConnection(
430 Server,
431 controllerPort,
432 settings.ServerSocketWorkerPoolSize,
```

testconductor/Conductor.scala, line 433 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: Controller()
File: testconductor/Conductor.scala:433
Taint Flags:

```
430 Server,
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.remote.testconductor

testconductor/Conductor.scala, line 433 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
431 controllerPort,
432 settings.ServerSocketWorkerPoolSize,
433 new ConductorHandler(settings.QueryTimeout, self, Logging(context.system, classOf[ConductorHandler])))
434
435 /*
436 * Supervision of the BarrierCoordinator means to catch all his bad emotions
```

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 238 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: selfPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:238
Taint Flags:

```
235 require(selfIndex >= 0 && selfIndex < maxNodes, "multinode.index is out of bounds: " + selfIndex)
236
237 private[testkit] val nodeConfig = mapToConfig(
238 Map(
239 "akka.actor.provider" -> "remote",
240 "akka.remote.artery.canonical.hostname" -> selfName,
241 "akka.remote.classic.netty.tcp.hostname" -> selfName,
```

testkit/MultiNodeSpec.scala, line 221 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serverPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:221
Taint Flags:

```
218 */
219 val serverPort: Int = Integer.getInteger("multinode.server-port", 4711)
220
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 221 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
221 require(serverPort > 0 && serverPort < 65535, "multinode.server-port is out of bounds: " + serverPort)
222
223 /**
224 * Index of this node in the roles sequence. The TestConductor
```

testkit/MultiNodeSpec.scala, line 221 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serverPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:221
Taint Flags:

```
218 */
219 val serverPort: Int = Integer.getInteger("multinode.server-port", 4711)
220
221 require(serverPort > 0 && serverPort < 65535, "multinode.server-port is out of bounds: " + serverPort)
222
223 /**
224 * Index of this node in the roles sequence. The TestConductor
```

testkit/MultiNodeSpec.scala, line 237 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: mapToConfig
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:237
Taint Flags:

```
234
235 require(selfIndex >= 0 && selfIndex < maxNodes, "multinode.index is out of bounds: " + selfIndex)
236
237 private[testkit] val nodeConfig = mapToConfig(
238 Map(
239 "akka.actor.provider" -> "remote",
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 237 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
240 "akka.remote.artery.canonical.hostname" -> selfName,
```

testkit/MultiNodeSpec.scala, line 468 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serverName
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:468
Taint Flags:

```
465 * Implementation (i.e. wait for start etc.)
466 */
467
468 private val controllerAddr = new InetSocketAddress(serverName, serverPort)
469
470 protected def attachConductor(tc: TestConductorExt): Unit = {
471 val timeout = tc.Settings.BarrierTimeout.duration
```

testkit/MultiNodeSpec.scala, line 141 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: maxNodes
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:141
Taint Flags:

```
138 val maxNodes: Int = Option(Integer.getInteger("multinode.max-nodes"))
139 .getOrElse(throw new IllegalStateException("need system property multinode.max-nodes to be set"))
140
141 require(maxNodes > 0, "multinode.max-nodes must be greater than 0")
142
143 /**
144 * Name (or IP address; must be resolvable using InetAddress.getByName)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 235 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: maxNodes
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:235
Taint Flags:

```
232 val selfIndex = Option(Integer.getInteger("multinode.index"))
233 .getOrElse(throw new IllegalStateException("need system property multinode.index to be set"))
234
235 require(selfIndex >= 0 && selfIndex < maxNodes, "multinode.index is out of bounds: " + selfIndex)
236
237 private[testkit] val nodeConfig = mapToConfig(
238 Map(
```

testkit/MultiNodeSpec.scala, line 185 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: udpPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:185
Taint Flags:

```
182 Integer.getInteger("multinode.udp.port", 0)
183 }
184
185 require(udpPort.getOrElse(1) >= 0 && udpPort.getOrElse(1) < 65535, "multinode.udp.port is out of bounds: " + udpPort)
186
187 /**
188 * Port number of this node.
```

testkit/MultiNodeSpec.scala, line 185 (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.remote.testkit

testkit/MultiNodeSpec.scala, line 185 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: udpPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:185
Taint Flags:

```
182 Integer.getInteger("multinode.udp.port", 0)
183 }
184
185 require(udpPort.getOrElse(1) >= 0 && udpPort.getOrElse(1) < 65535, "multinode.udp.port is out of bounds: " + udpPort)
186
187 /**
188 * Port number of this node.
```

testkit/MultiNodeSpec.scala, line 195 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: udpPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:195
Taint Flags:

```
192 */
193 val selfPort: Int =
194 System.getProperty("multinode.protocol") match {
195 case "udp" => udpPort.getOrElse(0)
196 case _ => tcpPort
197 }
198
```

testkit/MultiNodeSpec.scala, line 210 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serverName
Enclosing Method: MultiNodeSpec()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 210 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: testkit/MultiNodeSpec.scala:210
Taint Flags:

```
207 val serverName: String = Option(System.getProperty("multinode.server-host"))
208 .getOrElse(throw new IllegalStateException("need system property multinode.server-host to be set"))
209
210 require(serverName != "", "multinode.server-host must not be empty")
211
212 /**
213 * Port number of the node that's running the server system. Defaults to 4711.
```

testkit/MultiNodeSpec.scala, line 403 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: maxNodes
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:403
Taint Flags:

```
400 require(
401 initialParticipants > 0,
402 "initialParticipants must be a 'def' or early initializer, and it must be greater zero")
403 require(initialParticipants <= maxNodes, "not enough nodes to run this test")
404
405 /**
406 * Access to the barriers, failure injection, etc. The extension will have
```

testkit/MultiNodeSpec.scala, line 468 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: serverPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:468
Taint Flags:

```
465 * Implementation (i.e. wait for start etc.)
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 468 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
466  */
467
468  private val controllerAddr = new InetSocketAddress(serverName, serverPort)
469
470  protected def attachConductor(tc: TestConductorExt): Unit = {
471  val timeout = tc.Settings.BarrierTimeout.duration
```

testkit/MultiNodeSpec.scala, line 160 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: selfName
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:160
Taint Flags:

```
157  case Some(host) => host
158  }
159
160  require(selfName != "", "multinode.host must not be empty")
161
162  /**
163  * TCP Port number to be used when running tests on TCP. 0 means a random port.
```

testkit/MultiNodeSpec.scala, line 238 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: selfName
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:238
Taint Flags:

```
235  require(selfIndex >= 0 && selfIndex < maxNodes, "multinode.index is out of bounds: " + selfIndex)
236
237  private[testkit] val nodeConfig = mapToConfig(
238  Map(
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 238 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
239 "akka.actor.provider" -> "remote",
240 "akka.remote.artery.canonical.hostname" -> selfName,
241 "akka.remote.classic.netty.tcp.hostname" -> selfName,
```

testkit/MultiNodeSpec.scala, line 238 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: selfName
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:238
Taint Flags:

```
235 require(selfIndex >= 0 && selfIndex < maxNodes, "multinode.index is out of bounds: " + selfIndex)
236
237 private[testkit] val nodeConfig = mapToConfig(
238 Map(
239 "akka.actor.provider" -> "remote",
240 "akka.remote.artery.canonical.hostname" -> selfName,
241 "akka.remote.classic.netty.tcp.hostname" -> selfName,
```

testkit/MultiNodeSpec.scala, line 171 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: tcpPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:171
Taint Flags:

```
168 */
169 val tcpPort: Int = Integer.getInteger("multinode.port", 0)
170
171 require(tcpPort >= 0 && tcpPort < 65535, "multinode.port is out of bounds: " + tcpPort)
172
173 /**
174 * UDP Port number to be used when running tests on UDP. 0 means a random port.
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 171 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: tcpPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:171
Taint Flags:

```
168 */
169 val tcpPort: Int = Integer.getInteger("multinode.port", 0)
170
171 require(tcpPort >= 0 && tcpPort < 65535, "multinode.port is out of bounds: " + tcpPort)
172
173 /**
174 * UDP Port number to be used when running tests on UDP. 0 means a random port.
```

testkit/MultiNodeSpec.scala, line 196 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: tcpPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:196
Taint Flags:

```
193 val selfPort: Int =
194 System.getProperty("multinode.protocol") match {
195 case "udp" => udpPort.getOrElse(0)
196 case _ => tcpPort
197 }
198
199 /**
```

testkit/MultiNodeSpec.scala, line 238 (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.remote.testkit

testkit/MultiNodeSpec.scala, line 238 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: tcpPort
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:238
Taint Flags:

```
235 require(selfIndex >= 0 && selfIndex < maxNodes, "multinode.index is out of bounds: " + selfIndex)
236
237 private[testkit] val nodeConfig = mapToConfig(
238 Map(
239 "akka.actor.provider" -> "remote",
240 "akka.remote.artery.canonical.hostname" -> selfName,
241 "akka.remote.classic.netty.tcp.hostname" -> selfName,
```

testkit/MultiNodeSpec.scala, line 235 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: selfIndex
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:235
Taint Flags:

```
232 val selfIndex = Option(Integer.getInteger("multinode.index"))
233 .getOrElse(throw new IllegalStateException("need system property multinode.index to be set"))
234
235 require(selfIndex >= 0 && selfIndex < maxNodes, "multinode.index is out of bounds: " + selfIndex)
236
237 private[testkit] val nodeConfig = mapToConfig(
238 Map(
```

testkit/MultiNodeSpec.scala, line 235 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: selfIndex
Enclosing Method: MultiNodeSpec()

Code Correctness: Constructor Invokes Overridable Function	Low
Package: akka.remote.testkit	
testkit/MultiNodeSpec.scala, line 235 (Code Correctness: Constructor Invokes Overridable Function)	**Low**

File: testkit/MultiNodeSpec.scala:235
Taint Flags:

```
232 val selfIndex = Option(Integer.getInteger("multinode.index"))
233 .getOrElse(throw new IllegalStateException("need system property multinode.index to be set"))
234
235 require(selfIndex >= 0 && selfIndex < maxNodes, "multinode.index is out of bounds: " + selfIndex)
236
237 private[testkit] val nodeConfig = mapToConfig(
238 Map(
```

Code Correctness: Erroneous String Compare (1 issue)

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");
  }
```

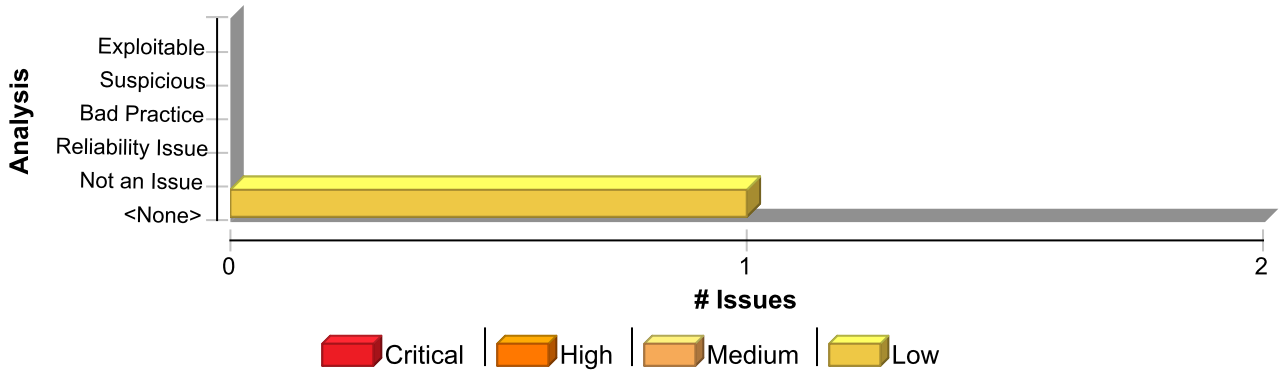
The `==` and `!=` operators will only behave as expected when they are used to compare strings contained in objects that are equal. The most common way for this to occur is for the strings to be interned, whereby the strings are added to a pool of objects maintained by the `String` class. Once a string 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	1	0	0	1
Total	**1**	**0**	**0**	**1**

Code Correctness: Erroneous String Compare — Low

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 194 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:194
Taint Flags:

```
191 * If set to 'udp', udpPort will be used. If unset or any other value, it will default to tcpPort.
192 */
193 val selfPort: Int =
194 System.getProperty("multinode.protocol") match {
195 case "udp" => udpPort.getOrElse(0)
196 case _ => tcpPort
197 }
```

Code Correctness: Non-Static Inner Class Implements Serializable (16 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

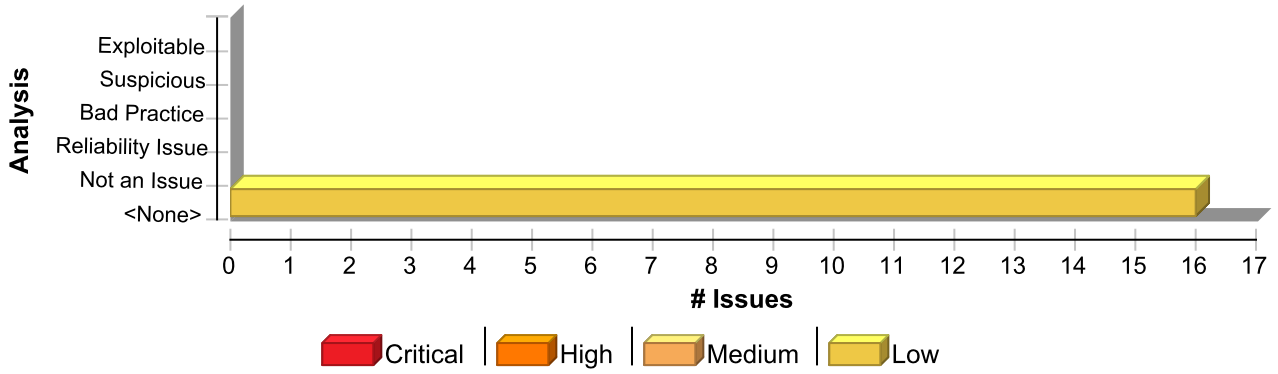
When using inner classes, they should not be serialized, or they should be changed to static-nested classes, since these do not have the drawbacks that non-static inner classes have when serialized. When a nested class is static it inherently has no association with instance variables (including those 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	16	0	0	16
Total	**16**	**0**	**0**	**16**

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

Package: akka.remote.testconductor

testconductor/Conductor.scala, line 537 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BarrierCoordinator$Data
File: testconductor/Conductor.scala:537
Taint Flags:

```
534
535 final case class RemoveClient(name: RoleName)
536
537 final case class Data(clients: Set[Controller.NodeInfo], barrier: String, arrived: List[ActorRef], deadline: Deadline)
538
539 trait Printer { this: Product with Throwable with NoStackTrace =>
540 override def toString = productPrefix + productIterator.mkString("(", ", ", ")")
```

testconductor/Player.scala, line 154 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ClientFSM$ConnectionFailure

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

Package: akka.remote.testconductor

testconductor/Player.scala, line 154 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

File: testconductor/Player.scala:154
Taint Flags:

```
151 final case class Data(channel: Option[Channel], runningOp: Option[(String, ActorRef)])
152
153 final case class Connected(channel: Channel) extends NoSerializationVerificationNeeded
154 final case class ConnectionFailure(msg: String) extends RuntimeException(msg) with NoStackTrace
155 case object Disconnected
156 }
157
```

testconductor/Conductor.scala, line 547 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BarrierCoordinator$FailedBarrier
File: testconductor/Conductor.scala:547
Taint Flags:

```
544 extends RuntimeException("timeout while waiting for barrier '" + data.barrier + "'")
545 with NoStackTrace
546 with Printer
547 final case class FailedBarrier(data: Data)
548 extends RuntimeException("failing barrier '" + data.barrier + "'")
549 with NoStackTrace
550 with Printer
```

testconductor/Conductor.scala, line 562 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BarrierCoordinator$ClientLost
File: testconductor/Conductor.scala:562
Taint Flags:

```
559 with NoStackTrace
560 with Printer
561 final case class BarrierEmpty(data: Data, msg: String) extends RuntimeException(msg) with NoStackTrace with Printer
```

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

Package: akka.remote.testconductor

testconductor/Conductor.scala, line 562 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
562 final case class ClientLost(data: Data, client: RoleName)
563 extends RuntimeException("unannounced disconnect of " + client)
564 with NoStackTrace
565 with Printer
```

testconductor/Conductor.scala, line 555 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BarrierCoordinator$WrongBarrier
File: testconductor/Conductor.scala:555
Taint Flags:

```
552 extends RuntimeException(node.toString)
553 with NoStackTrace
554 with Printer
555 final case class WrongBarrier(barrier: String, client: ActorRef, data: Data)
556 extends RuntimeException(
557 data.clients.find(_.fsm == client).map(_.name.toString).getOrElse(client.toString) +
558 " tried to enter '" + barrier + "' while we were waiting for '" + data.barrier + "'")
```

testconductor/Conductor.scala, line 412 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Controller$CreateServerFSM
File: testconductor/Conductor.scala:412
Taint Flags:

```
409 class ClientDisconnectedException(msg: String) extends AkkaException(msg) with NoStackTrace
410 case object GetNodes
411 case object GetSockAddr
412 final case class CreateServerFSM(channel: Channel) extends NoSerializationVerificationNeeded
413
414 final case class NodeInfo(name: RoleName, addr: Address, fsm: ActorRef)
415 }
```

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

Package: akka.remote.testconductor

testconductor/Conductor.scala, line 408 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Controller$ClientDisconnected
File: testconductor/Conductor.scala:408
Taint Flags:

```
405 * INTERNAL API.
406 */
407 private[akka] object Controller {
408 final case class ClientDisconnected(name: RoleName) extends DeadLetterSuppression
409 class ClientDisconnectedException(msg: String) extends AkkaException(msg) with NoStackTrace
410 case object GetNodes
411 case object GetSockAddr
```

testconductor/Conductor.scala, line 535 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BarrierCoordinator$RemoveClient
File: testconductor/Conductor.scala:535
Taint Flags:

```
532 case object Idle extends State
533 case object Waiting extends State
534
535 final case class RemoveClient(name: RoleName)
536
537 final case class Data(clients: Set[Controller.NodeInfo], barrier: String, arrived: List[ActorRef], deadline: Deadline)
538
```

testconductor/Player.scala, line 151 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

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

Package: akka.remote.testconductor

testconductor/Player.scala, line 151 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: ClientFSM$Data
File: testconductor/Player.scala:151
Taint Flags:

```
148 case object Connected extends State
149 case object Failed extends State
150
151 final case class Data(channel: Option[Channel], runningOp: Option[(String, ActorRef)])
152
153 final case class Connected(channel: Channel) extends NoSerializationVerificationNeeded
154 final case class ConnectionFailure(msg: String) extends RuntimeException(msg) with NoStackTrace
```

testconductor/Conductor.scala, line 414 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Controller$NodeInfo
File: testconductor/Conductor.scala:414
Taint Flags:

```
411 case object GetSockAddr
412 final case class CreateServerFSM(channel: Channel) extends NoSerializationVerificationNeeded
413
414 final case class NodeInfo(name: RoleName, addr: Address, fsm: ActorRef)
415 }
416
417 /**
```

testconductor/Player.scala, line 153 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ClientFSM$Connected
File: testconductor/Player.scala:153
Taint Flags:

```
150
151 final case class Data(channel: Option[Channel], runningOp: Option[(String, ActorRef)])
```

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

Package: akka.remote.testconductor

testconductor/Player.scala, line 153 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
152
153 final case class Connected(channel: Channel) extends NoSerializationVerificationNeeded
154 final case class ConnectionFailure(msg: String) extends RuntimeException(msg) with NoStackTrace
155 case object Disconnected
156 }
```

testconductor/Conductor.scala, line 409 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: Controller$ClientDisconnectedException
File: testconductor/Conductor.scala:409
Taint Flags:

```
406 */
407 private[akka] object Controller {
408 final case class ClientDisconnected(name: RoleName) extends DeadLetterSuppression
409 class ClientDisconnectedException(msg: String) extends AkkaException(msg) with NoStackTrace
410 case object GetNodes
411 case object GetSockAddr
412 final case class CreateServerFSM(channel: Channel) extends NoSerializationVerificationNeeded
```

testconductor/Conductor.scala, line 543 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BarrierCoordinator$BarrierTimeout
File: testconductor/Conductor.scala:543
Taint Flags:

```
540 override def toString = productPrefix + productIterator.mkString("(", ", ", ")")
541 }
542
543 final case class BarrierTimeout(data: Data)
544 extends RuntimeException("timeout while waiting for barrier '" + data.barrier + "'")
545 with NoStackTrace
546 with Printer
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.remote.testconductor

testconductor/Conductor.scala, line 543 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

testconductor/Conductor.scala, line 551 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BarrierCoordinator$DuplicateNode
File: testconductor/Conductor.scala:551
Taint Flags:

```
548 extends RuntimeException("failing barrier '" + data.barrier + "'")
549 with NoStackTrace
550 with Printer
551 final case class DuplicateNode(data: Data, node: Controller.NodeInfo)
552 extends RuntimeException(node.toString)
553 with NoStackTrace
554 with Printer
```

testconductor/Conductor.scala, line 561 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: BarrierCoordinator$BarrierEmpty
File: testconductor/Conductor.scala:561
Taint Flags:

```
558 " tried to enter '" + barrier + "' while we were waiting for '" + data.barrier + "'")
559 with NoStackTrace
560 with Printer
561 final case class BarrierEmpty(data: Data, msg: String) extends RuntimeException(msg) with NoStackTrace with Printer
562 final case class ClientLost(data: Data, client: RoleName)
563 extends RuntimeException("unannounced disconnect of " + client)
564 with NoStackTrace
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low
Package: akka.remote.testkit	
testkit/MultiNodeSpec.scala, line 487 (Code Correctness: Non-Static Inner Class Implements Serializable)	**Low**

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: MultiNodeSpec$Replacement
File: testkit/MultiNodeSpec.scala:487
Taint Flags:

```
484 // now add deployments, if so desired
485
486 // Cannot be final because of https://github.com/scala/bug/issues/4440
487 private case class Replacement(tag: String, role: RoleName) {
488 lazy val addr = node(role).address.toString
489 }
490
```

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`.

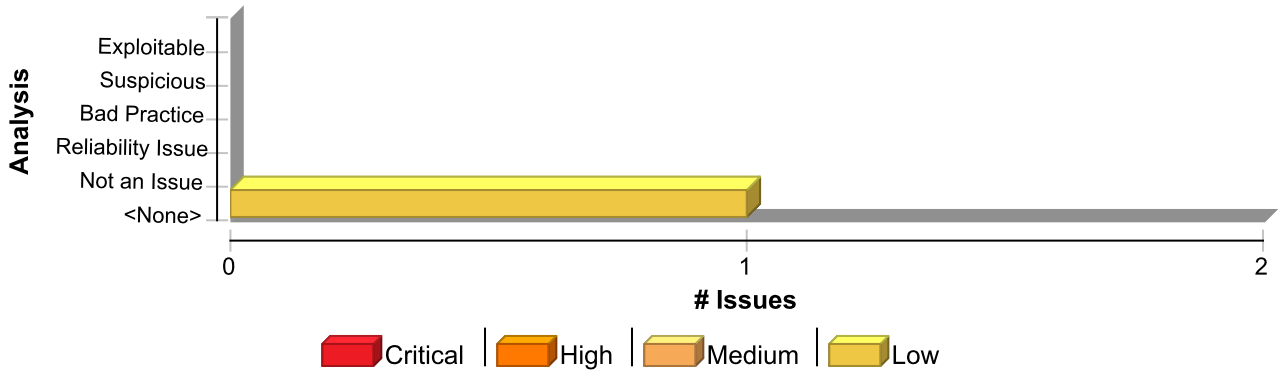
```
public void setUpCalls() {
  boolean firstCall = false;
  boolean secondCall = false;

  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.remote.testkit

testkit/MultiNodeSpec.scala, line 156 (Dead Code: Expression is Always false) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: IfStatement
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:156
Taint Flags:

```
153 */
154 val selfName: String = Option(System.getProperty("multinode.host")) match {
155 case None => throw new IllegalStateException("need system property multinode.host to be set")
156 case Some("") => InetAddress.getLocalHost.getHostAddress
157 case Some(host) => host
158 }
159
```

J2EE Bad Practices: JVM Termination (1 issue)

Abstract

A web application should not attempt to shut down its container.

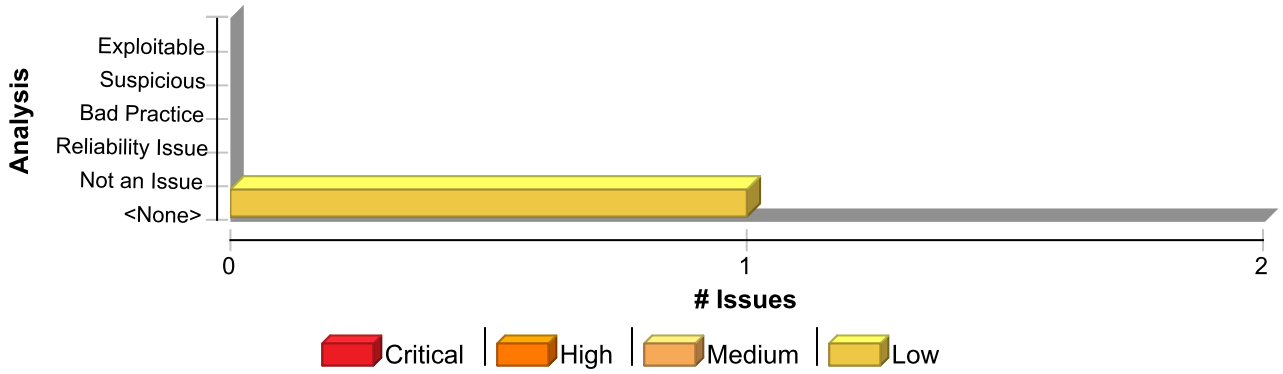
Explanation

It is never a good idea for a web application to attempt to shut down the application container. A call to a termination method is probably part of leftover debug code or code imported from a non-J2EE application.

Recommendation

Never call a termination method within a web application. Such method calls in a J2EE application indicates poor software hygiene and should be removed. Regardless of whether there is a perceived threat, it is unlikely that there is a legitimate reason for such code to remain in the application.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: JVM Termination	1	0	0	1
Total	**1**	**0**	**0**	**1**

J2EE Bad Practices: JVM Termination	Low
Package: akka.remote.testconductor	
testconductor/Player.scala, line 291 (J2EE Bad Practices: JVM Termination)	**Low**

Issue Details

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

Sink Details

Sink: exit()
Enclosing Method: applyOrElse()
File: testconductor/Player.scala:291
Taint Flags:

J2EE Bad Practices: JVM Termination	Low

Package: akka.remote.testconductor

testconductor/Player.scala, line 291 (J2EE Bad Practices: JVM Termination)	Low

```
288 context.system.asInstanceOf[ActorSystemImpl].abort()
289 stop()
290 case TerminateMsg(Right(exitValue)) =>
291 System.exit(exitValue)
292 stay() // needed because Java doesn’t have Nothing
293 case _: Done => stay() //FIXME what should happen?
294 }
```

J2EE Bad Practices: Sockets (1 issue)

Abstract

Socket-based communication in web applications is prone to error.

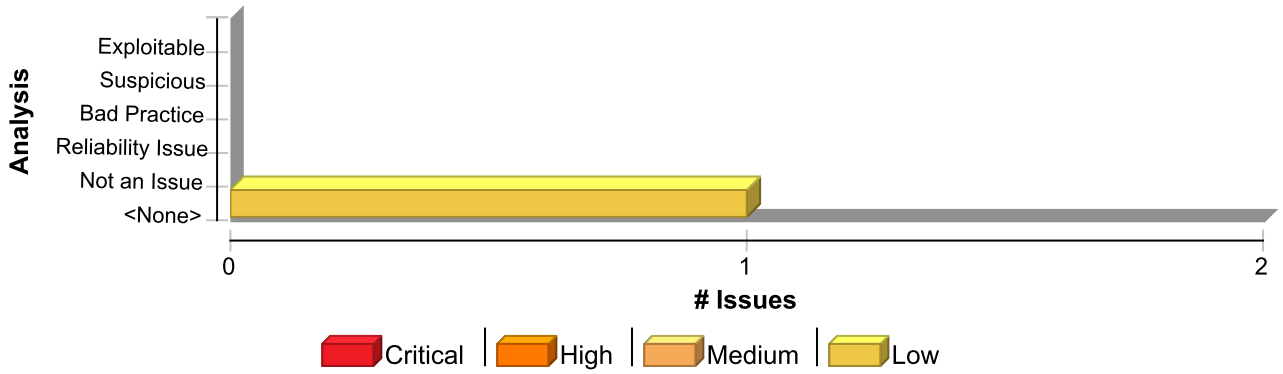
Explanation

The J2EE standard permits the use of sockets only for the purpose of communication with legacy systems when no higher-level protocol is available. Authoring your own communication protocol requires wrestling with difficult security issues, including: - In-band versus out-of-band signaling - Compatibility between protocol versions - Channel security - Error handling - Network constraints (firewalls) - Session management Without significant scrutiny by a security expert, chances are good that a custom communication protocol will suffer from security problems. Many of the same issues apply to a custom implementation of a standard protocol. While there are usually more resources available that address security concerns related to implementing a standard protocol, these resources are also available to attackers.

Recommendation

Replace a custom communication protocol with an industry standard protocol or framework. Consider whether you can use a protocol such as HTTP, FTP, SMTP, CORBA, RMI/IIOP, EJB, or SOAP. Consider the security track record of the protocol implementation you choose.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
J2EE Bad Practices: Sockets	1	0	0	1
Total	**1**	**0**	**0**	**1**

J2EE Bad Practices: Sockets	Low
Package: akka.remote.testkit	
testkit/MultiNodeSpec.scala, line 468 (J2EE Bad Practices: Sockets)	**Low**

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

J2EE Bad Practices: Sockets	Low

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 468 (J2EE Bad Practices: Sockets) — Low

Sink: InetSocketAddress()
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:468
Taint Flags:

```
465 * Implementation (i.e. wait for start etc.)
466 */
467
468 private val controllerAddr = new InetSocketAddress(serverName, serverPort)
469
470 protected def attachConductor(tc: TestConductorExt): Unit = {
471 val timeout = tc.Settings.BarrierTimeout.duration
```

Often Misused: Authentication (3 issues)

Abstract

Attackers may spoof DNS entries. Do not rely on DNS names for security.

Explanation

Many DNS servers are susceptible to spoofing attacks, so you should assume that your software will someday run in an environment with a compromised DNS server. If attackers are allowed to make DNS updates (sometimes called DNS cache poisoning), they can route your network traffic through their machines or make it appear as if their IP addresses are part of your domain. Do not base the security of your system on DNS names. **Example:** The following code uses a DNS lookup to determine whether an inbound request is from a trusted host. If an attacker can poison the DNS cache, they can gain trusted status.

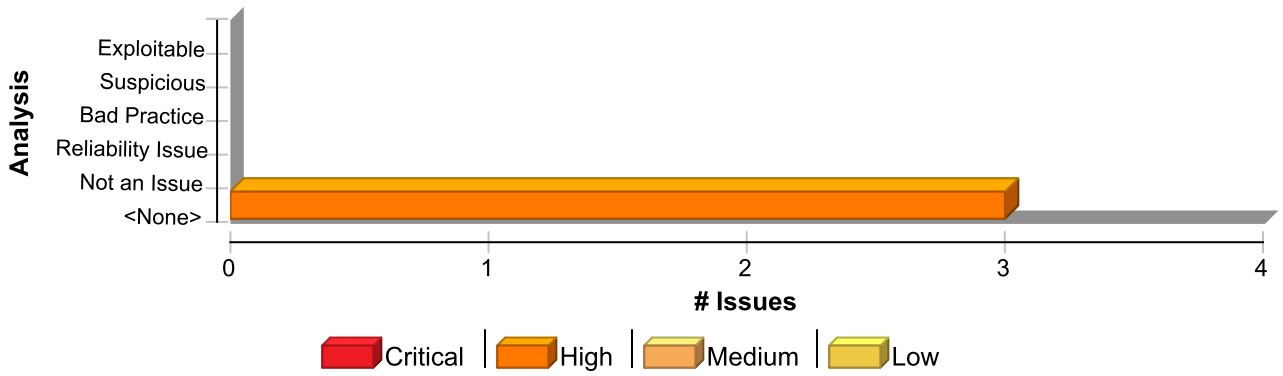
```
String ip = request.getRemoteAddr();
InetAddress addr = InetAddress.getByName(ip);
if (addr.getCanonicalHostName().endsWith("trustme.com")) {
trusted = true;
}
```

IP addresses are more reliable than DNS names, but they can also be spoofed. Attackers may easily forge the source IP address of the packets they send, but response packets will return to the forged IP address. To see the response packets, the attacker has to sniff the traffic between the victim machine and the forged IP address. In order to accomplish the required sniffing, attackers typically attempt to locate themselves on the same subnet as the victim machine. Attackers may be able to circumvent this requirement by using source routing, but source routing is disabled across much of the Internet today. In summary, IP address verification can be a useful part of an authentication scheme, but it should not be the single factor required for authentication.

Recommendation

You can increase confidence in a domain name lookup if you check to make sure that the host's forward and backward DNS entries match. Attackers will not be able to spoof both the forward and the reverse DNS entries without controlling the nameservers for the target domain. This is not a foolproof approach however: attackers may be able to convince the domain registrar to turn over the domain to a malicious nameserver. Basing authentication on DNS entries is simply a risky proposition. While no authentication mechanism is foolproof, there are better alternatives than host-based authentication. Password systems offer decent security, but are susceptible to bad password choices, insecure password transmission, and bad password management. A cryptographic scheme like SSL is worth considering, but such schemes are often so complex that they bring with them the risk of significant implementation errors, and key material can always be stolen. In many situations, multi-factor authentication including a physical token offers the most security available at a reasonable price.

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Often Misused: Authentication	3	0	0	3
Total	**3**	**0**	**0**	**3**

Often Misused: Authentication — High

Package: akka.remote.testconductor

testconductor/Conductor.scala, line 466 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getHostAddress()
Enclosing Method: applyOrElse()
File: testconductor/Conductor.scala:466
Taint Flags:

```
463 override def receive = LoggingReceive {
464 case CreateServerFSM(channel) =>
465 val (ip, port) = channel.getRemoteAddress match {
466 case s: InetSocketAddress => (s.getAddress.getHostAddress, s.getPort)
467 case _ => throw new RuntimeException() // compiler exhaustiveness check pleaser
468 }
469 val name = ip + ":" + port + "-server" + generation.next()
```

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 156 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getLocalHost()
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:156
Taint Flags:

```
153 */
154 val selfName: String = Option(System.getProperty("multinode.host")) match {
155 case None => throw new IllegalStateException("need system property multinode.host to be set")
156 case Some("") => InetAddress.getLocalHost.getHostAddress
157 case Some(host) => host
158 }
159
```

Often Misused: Authentication — High

Package: akka.remote.testkit

testkit/MultiNodeSpec.scala, line 156 (Often Misused: Authentication) — High

Issue Details

Kingdom: API Abuse
Scan Engine: SCA (Semantic)

Sink Details

Sink: getHostAddress()
Enclosing Method: MultiNodeSpec()
File: testkit/MultiNodeSpec.scala:156
Taint Flags:

```
153 */
154 val selfName: String = Option(System.getProperty("multinode.host")) match {
155 case None => throw new IllegalStateException("need system property multinode.host to be set")
156 case Some("") => InetAddress.getLocalHost.getHostAddress
157 case Some(host) => host
158 }
159
```

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:

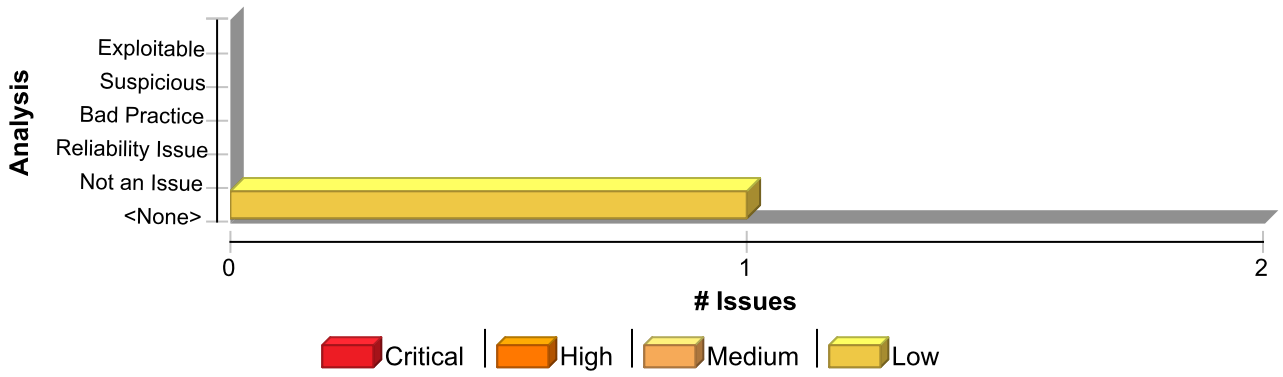
```
...
println(Properties.osName)
...
```

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: testkit

testkit/PerfFlamesSupport.scala, line 36 (System Information Leak: Internal) — Low

Issue Details

Kingdom: Encapsulation
Scan Engine: SCA (Data Flow)

Source Details

Source: java.lang.management.RuntimeMXBean.getName()
From: akka.remote.testkit.PerfFlamesSupport$anonfun$runPerfFlames$2.apply
File: testkit/PerfFlamesSupport.scala:32

```
29 val afterDelay = akka.pattern.after(delay, system.scheduler)(Future.successful("GO!"))
30 afterDelay.onComplete { _ =>
31 import java.lang.management._
32 val name = ManagementFactory.getRuntimeMXBean.getName
33 val pid = name.substring(0, name.indexOf('@')).toInt
34
35 val perfCommand = s"$perfJavaFlamesPath $pid"
```

Sink Details

Sink: scala.Predef.println()
Enclosing Method: apply()
File: testkit/PerfFlamesSupport.scala:36
Taint Flags: NUMBER, PRIMARY_KEY, SYSTEMINFO

```
33 val pid = name.substring(0, name.indexOf('@')).toInt
34
35 val perfCommand = s"$perfJavaFlamesPath $pid"
36 println(s"[perf @ $myself($pid)][OUT]: " + perfCommand)
37
38 import scala.sys.process._
39 perfCommand.run(new ProcessLogger {
```