Fortify Standalone Report Generator

Developer Workbook

akka-coordination

Table of Contents

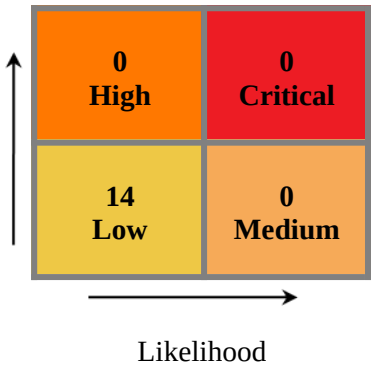
Executive Summary

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-coordination 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-coordination
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, 11:25 AM	**Engine Version:**	21.1.1.0009
Host Name:	Jacks-Work-MBP.local	**Certification:**	VALID
Number of Files:	13	**Lines of Code:**	298

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 / 5	0 / 5
Code Correctness: Non-Static Inner Class Implements Serializable	0	0	0	0 / 9	0 / 9

Results Outline

Code Correctness: Constructor Invokes Overridable Function (5 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");
    }
  }
}
```

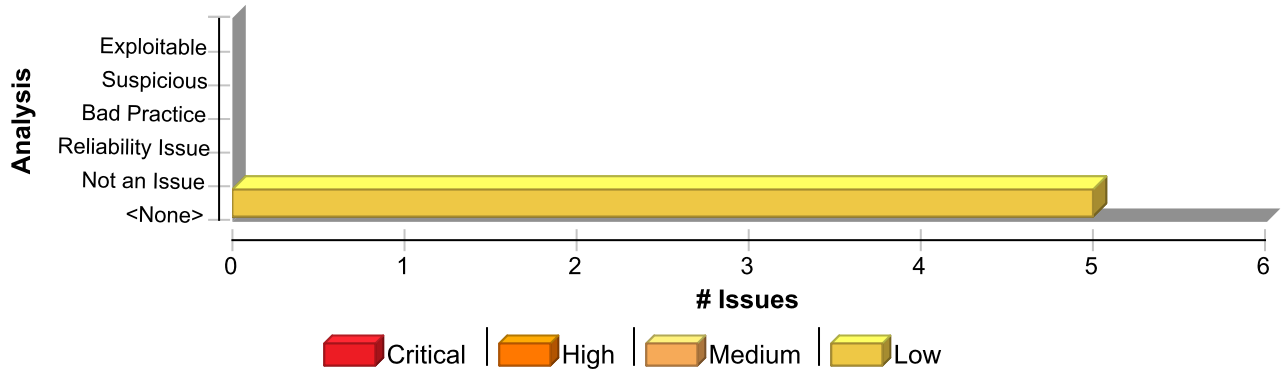
The code in `Example 1` prints "Attack successful!", since the `Attacker` class overrides the `validateUser()` function that is called from the constructor of the superclass `User`, and Java will first look in the subclass for functions called from the constructor.

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

Code Correctness: Constructor Invokes Overridable Function	**Low**
Package: akka.coordination.lease	
test/scala/akka/coordination/lease/TestLeaseActor.scala, line 105 (Code Correctness: Constructor Invokes Overridable Function)	**Low**
Issue Details	

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.coordination.lease

test/scala/akka/coordination/lease/TestLeaseActor.scala, line 105 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: leaseActor
Enclosing Method: TestLeaseActorClient()
File: test/scala/akka/coordination/lease/TestLeaseActor.scala:105
Taint Flags:

```
102 val leaseActor = TestLeaseActorClientExt(system).getLeaseActor()
103
104 log.info("lease created {}", settings)
105 leaseActor ! Create(settings.leaseName, settings.ownerName)
106
107 private implicit val timeout: Timeout = Timeout(100.seconds)
108
```

test/scala/akka/coordination/lease/TestLease.scala, line 73 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: log
Enclosing Method: TestLease()
File: test/scala/akka/coordination/lease/TestLease.scala:73
Taint Flags:

```
70 private val nextCheckLeaseResult = new AtomicReference[Boolean](false)
71 private val currentCallBack = new AtomicReference[Option[Throwable] => Unit](_ => ())
72
73 log.info("Creating lease {}", settings)
74
75 TestLeaseExt(system).setTestLease(settings.leaseName, this)
76
```

test/scala/akka/coordination/lease/TestLeaseActor.scala, line 104 (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.coordination.lease

test/scala/akka/coordination/lease/TestLeaseActor.scala, line 104 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: log
Enclosing Method: TestLeaseActorClient()
File: test/scala/akka/coordination/lease/TestLeaseActor.scala:104
Taint Flags:

```
101 private val log = Logging(system, classOf[TestLeaseActorClient])
102 val leaseActor = TestLeaseActorClientExt(system).getLeaseActor()
103
104 log.info("lease created {}", settings)
105 leaseActor ! Create(settings.leaseName, settings.ownerName)
106
107 private implicit val timeout: Timeout = Timeout(100.seconds)
```

test/scala/akka/coordination/lease/TestLease.scala, line 69 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: initialPromise
Enclosing Method: TestLease()
File: test/scala/akka/coordination/lease/TestLease.scala:69
Taint Flags:

```
66
67 val initialPromise = Promise[Boolean]()
68
69 private val nextAcquireResult = new AtomicReference[Future[Boolean]](initialPromise.future)
70 private val nextCheckLeaseResult = new AtomicReference[Boolean](false)
71 private val currentCallBack = new AtomicReference[Option[Throwable] => Unit](_ => ())
72
```

Package: akka.coordination.lease.scaladsl

test/scala/akka/coordination/lease/scaladsl/LeaseProviderSpec.scala, line 66 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: config

Code Correctness: Constructor Invokes Overridable Function	Low
Package: akka.coordination.lease.scaladsl	
test/scala/akka/coordination/lease/scaladsl/LeaseProviderSpec.scala, line 66 (Code Correctness: Constructor Invokes Overridable Function)	**Low**

Enclosing Method: LeaseProviderSpec()
File: test/scala/akka/coordination/lease/scaladsl/LeaseProviderSpec.scala:66
Taint Flags:

```
63  """)
64 }
65
66 class LeaseProviderSpec extends AkkaSpec(LeaseProviderSpec.config) {
67  import LeaseProviderSpec._
68
69  "LeaseProvider" must {
```

Code Correctness: Non-Static Inner Class Implements Serializable (9 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

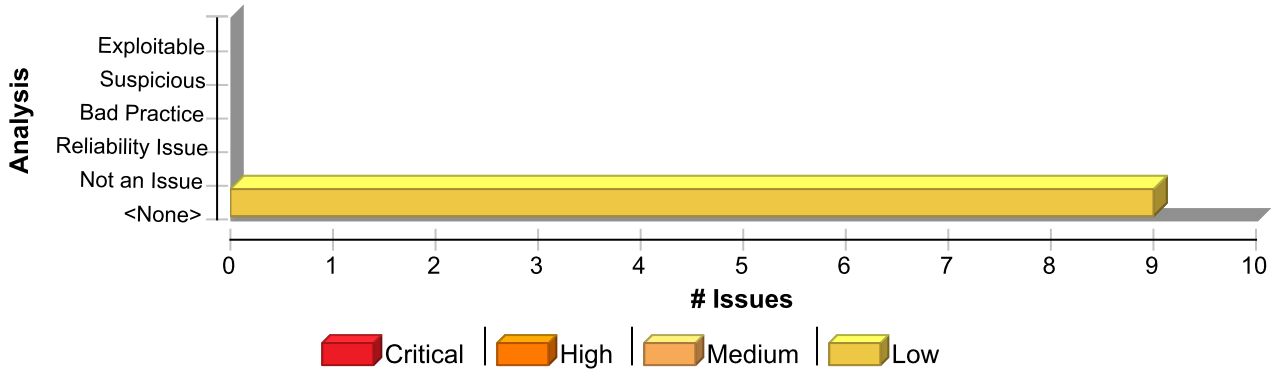
When using inner classes, they should not be serialized, or they should be changed to static-nested classes, since these do not have the drawbacks that non-static inner classes have when serialized. When a nested class is static it inherently has no association with instance variables (including those 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	9	0	0	9
Total	**9**	**0**	**0**	**9**

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

Package: akka.coordination.lease

test/scala/akka/coordination/lease/TestLease.scala, line 51 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TestLease$AcquireReq
File: test/scala/akka/coordination/lease/TestLease.scala:51
Taint Flags:

```
48 }
49
50 object TestLease {
51  final case class AcquireReq(owner: String)
52  final case class ReleaseReq(owner: String)
53
54  val config = ConfigFactory.parseString(s"""
```

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

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TestLeaseActor$Create

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

Package: akka.coordination.lease

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

File: test/scala/akka/coordination/lease/TestLeaseActor.scala:35
Taint Flags:

```
32 sealed trait LeaseRequest extends JavaSerializable
33 final case class Acquire(owner: String) extends LeaseRequest
34 final case class Release(owner: String) extends LeaseRequest
35 final case class Create(leaseName: String, ownerName: String) extends JavaSerializable
36
37 case object GetRequests extends JavaSerializable
38 final case class LeaseRequests(requests: List[LeaseRequest]) extends JavaSerializable
```

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

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TestLeaseActor$Acquire
File: test/scala/akka/coordination/lease/TestLeaseActor.scala:33
Taint Flags:

```
30 Props(new TestLeaseActor)
31
32 sealed trait LeaseRequest extends JavaSerializable
33 final case class Acquire(owner: String) extends LeaseRequest
34 final case class Release(owner: String) extends LeaseRequest
35 final case class Create(leaseName: String, ownerName: String) extends JavaSerializable
36
```

test/scala/akka/coordination/lease/TestLeaseActor.scala, line 39 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TestLeaseActor$ActionRequest
File: test/scala/akka/coordination/lease/TestLeaseActor.scala:39
Taint Flags:

```
36
37 case object GetRequests extends JavaSerializable
38 final case class LeaseRequests(requests: List[LeaseRequest]) extends JavaSerializable
```

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

Package: akka.coordination.lease

test/scala/akka/coordination/lease/TestLeaseActor.scala, line 39 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
39  final case class ActionRequest(request: LeaseRequest, result: Any) extends JavaSerializable // boolean of Failure
40 }
41
42 class TestLeaseActor extends Actor with ActorLogging {
```

test/scala/akka/coordination/lease/TestLease.scala, line 52 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TestLease$ReleaseReq
File: test/scala/akka/coordination/lease/TestLease.scala:52
Taint Flags:

```
49
50 object TestLease {
51  final case class AcquireReq(owner: String)
52  final case class ReleaseReq(owner: String)
53
54  val config = ConfigFactory.parseString(s"""
55  test-lease {
```

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

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TestLeaseActor$Release
File: test/scala/akka/coordination/lease/TestLeaseActor.scala:34
Taint Flags:

```
31
32  sealed trait LeaseRequest extends JavaSerializable
33  final case class Acquire(owner: String) extends LeaseRequest
34  final case class Release(owner: String) extends LeaseRequest
35  final case class Create(leaseName: String, ownerName: String) extends JavaSerializable
36
37  case object GetRequests extends JavaSerializable
```

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

Package: akka.coordination.lease

test/scala/akka/coordination/lease/TestLeaseActor.scala, line 38 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: TestLeaseActor$LeaseRequests
File: test/scala/akka/coordination/lease/TestLeaseActor.scala:38
Taint Flags:

```
35 final case class Create(leaseName: String, ownerName: String) extends JavaSerializable
36
37 case object GetRequests extends JavaSerializable
38 final case class LeaseRequests(requests: List[LeaseRequest]) extends JavaSerializable
39 final case class ActionRequest(request: LeaseRequest, result: Any) extends JavaSerializable // boolean of Failure
40 }
41
```

Package: akka.coordination.lease.javadsl

main/scala/akka/coordination/lease/javadsl/LeaseProvider.scala, line 21 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeaseProvider$LeaseKey
File: main/scala/akka/coordination/lease/javadsl/LeaseProvider.scala:21
Taint Flags:

```
18
19 override def createExtension(system: ExtendedActorSystem): LeaseProvider = new LeaseProvider(system)
20
21 private final case class LeaseKey(leaseName: String, configPath: String, clientName: String)
22 }
23
24 class LeaseProvider(system: ExtendedActorSystem) extends Extension {
```

Package: akka.coordination.lease.scaladsl

main/scala/akka/coordination/lease/scaladsl/LeaseProvider.scala, line 32 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality

Code Correctness: Non-Static Inner Class Implements Serializable	Low
Package: akka.coordination.lease.scaladsl	
main/scala/akka/coordination/lease/scaladsl/LeaseProvider.scala, line 32 (Code Correctness: Non-Static Inner Class Implements Serializable)	**Low**

Scan Engine: SCA (Structural)

Sink Details

Sink: Class: LeaseProvider$LeaseKey
File: main/scala/akka/coordination/lease/scaladsl/LeaseProvider.scala:32
Taint Flags:

```
29
30  override def createExtension(system: ExtendedActorSystem): LeaseProvider = new LeaseProvider(system)
31
32  private final case class LeaseKey(leaseName: String, configPath: String, clientName: String)
33 }
34
35 final class LeaseProvider(system: ExtendedActorSystem) extends Extension {
```