Fortify Standalone Report Generator

Developer Workbook

akka-cluster-typed

Table of Contents

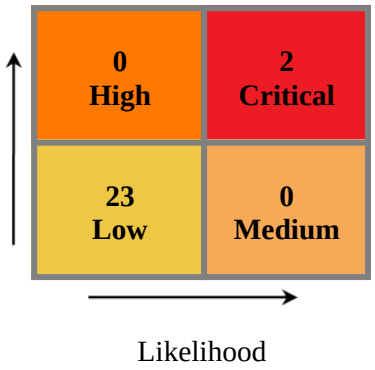
Executive Summary

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

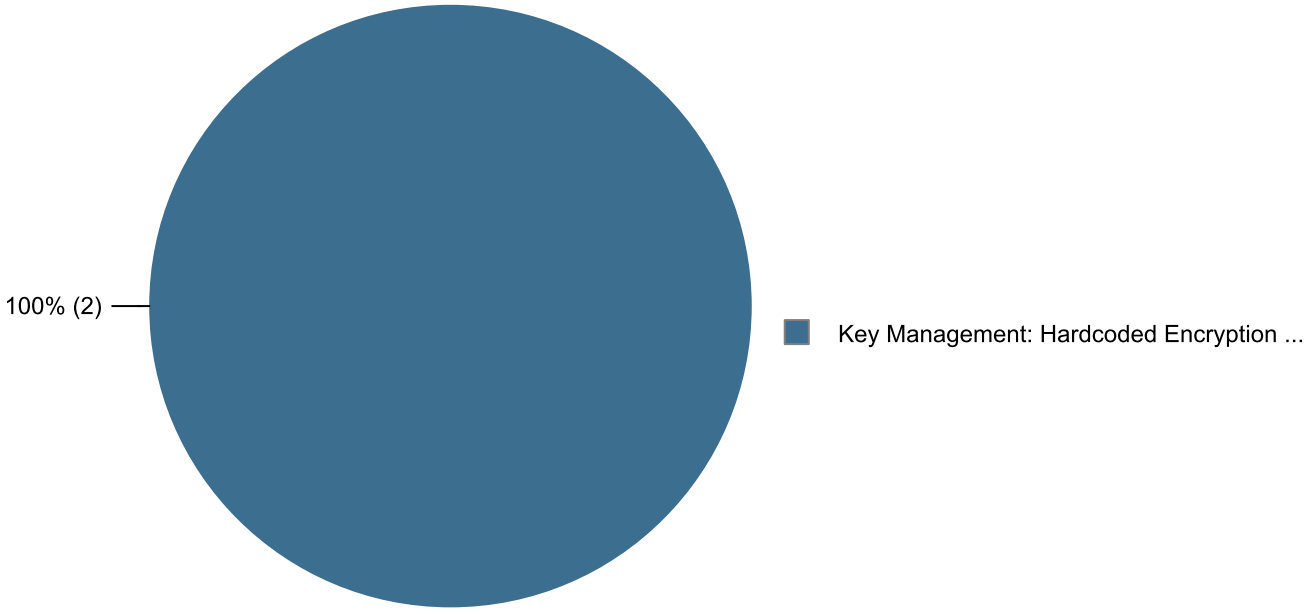
This section provides an overview of the issues uncovered during analysis.

Project Name:	akka-cluster-typed
Project Version:	
SCA:	Results Present
WebInspect:	Results Not Present
WebInspect Agent:	Results Not Present
Other:	Results Not Present

Issues by Priority



Top Ten Critical Categories



Project Description

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

SCA

Date of Last Analysis:	Jun 16, 2022, 11:25 AM	**Engine Version:**	21.1.1.0009
Host Name:	Jacks-Work-MBP.local	**Certification:**	VALID
Number of Files:	21	**Lines of Code:**	1,273

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 / 10	0 / 10
Code Correctness: Erroneous String Compare	0	0	0	0 / 3	0 / 3
Code Correctness: Non-Static Inner Class Implements Serializable	0	0	0	0 / 10	0 / 10
Key Management: Hardcoded Encryption Key	0 / 2	0	0	0	0 / 2

Results Outline

Code Correctness: Constructor Invokes Overridable Function (10 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");
    }
  }
}
```

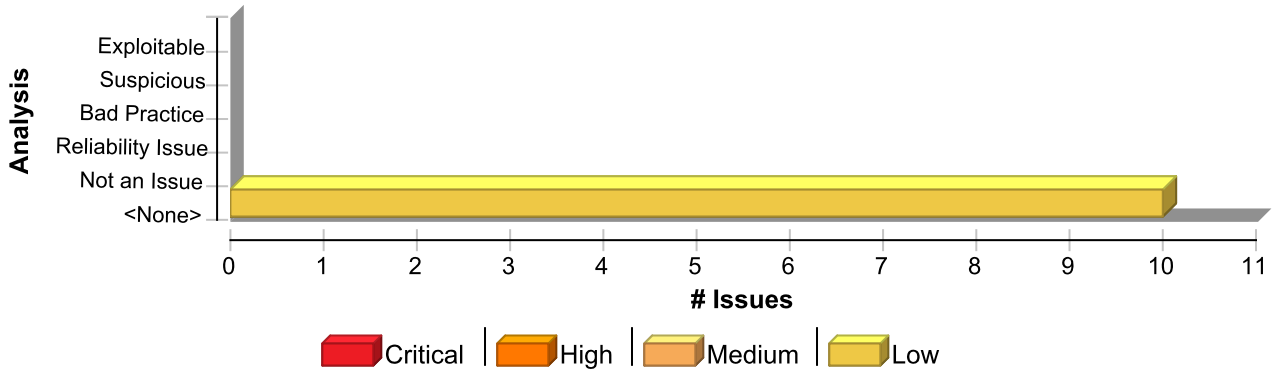
The code in `Example 1` prints "Attack successful!", since the `Attacker` class overrides the `validateUser()` function that is called from the constructor of the superclass `User`, and Java will first look in the subclass for functions called from the constructor.

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	10	0	0	10
Total	**10**	**0**	**0**	**10**

Code Correctness: Constructor Invokes Overridable Function	**Low**
Package: akka.cluster.ddata.typed.scaladsl	
scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 93 (Code Correctness: Constructor Invokes Overridable Function)	**Low**
Issue Details	

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata.typed.scaladsl

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 93 (Code Correctness: Constructor Invokes Overridable Function) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: DistributedData()
File: scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala:93
Taint Flags:

```
90  if (isTerminated) {
91  val log = LoggerFactory.getLogger(getClass)
92  if (Cluster(classicSystem).isTerminated)
93  log.warn("Replicator points to dead letters, because Cluster is terminated.")
94  else
95  log.warn2(
96  "Replicator points to dead letters. Make sure the cluster node has the proper role. " +
```

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 101 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: settings
Enclosing Method: DistributedData()
File: scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala:101
Taint Flags:

```
98  Cluster(classicSystem).selfRoles.mkString(","),
99  settings.roles.mkString(","))
100 system.deadLetters
101 } else {
102 val underlyingReplicator = dd.DistributedData(classicSystem).replicator
103 val replicatorBehavior = Replicator.behavior(settings, underlyingReplicator)
104
```

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 103 (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.typed.scaladsl

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 103 (Code Correctness: Constructor Invokes Overridable Function) — Low

Sink Details

Sink: FunctionCall: settings
Enclosing Method: DistributedData()
File: scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala:103
Taint Flags:

```
100 system.deadLetters
101 } else {
102 val underlyingReplicator = dd.DistributedData(classicSystem).replicator
103 val replicatorBehavior = Replicator.behavior(settings, underlyingReplicator)
104
105 system.internalSystemActorOf(
106 replicatorBehavior,
```

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 82 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: classicSystem
Enclosing Method: DistributedData()
File: scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala:82
Taint Flags:

```
79
80 private val classicSystem = system.toClassic.asInstanceOf[ExtendedActorSystem]
81
82 implicit val selfUniqueAddress: SelfUniqueAddress = dd.DistributedData(classicSystem).selfUniqueAddress
83
84 /**
85 * `ActorRef` of the [[Replicator]].
```

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: classicSystem
Enclosing Method: DistributedData()

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata.typed.scaladsl

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 92 (Code Correctness: Constructor Invokes Overridable Function) — Low

File: scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala:92
Taint Flags:

```
89 val replicator: ActorRef[Replicator.Command] =
90 if (isTerminated) {
91 val log = LoggerFactory.getLogger(getClass)
92 if (Cluster(classicSystem).isTerminated)
93 log.warn("Replicator points to dead letters, because Cluster is terminated.")
94 else
95 log.warn2(
```

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 93 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: classicSystem
Enclosing Method: DistributedData()
File: scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala:93
Taint Flags:

```
90 if (isTerminated) {
91 val log = LoggerFactory.getLogger(getClass)
92 if (Cluster(classicSystem).isTerminated)
93 log.warn("Replicator points to dead letters, because Cluster is terminated.")
94 else
95 log.warn2(
96 "Replicator points to dead letters. Make sure the cluster node has the proper role. " +
```

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 102 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: classicSystem
Enclosing Method: DistributedData()
File: scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala:102
Taint Flags:

```
99 settings.roles.mkString(","))
```

Code Correctness: Constructor Invokes Overridable Function — Low

Package: akka.cluster.ddata.typed.scaladsl

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 102 (Code Correctness: Constructor Invokes Overridable Function) — Low

```
100 system.deadLetters
101 } else {
102 val underlyingReplicator = dd.DistributedData(classicSystem).replicator
103 val replicatorBehavior = Replicator.behavior(settings, underlyingReplicator)
104
105 system.internalSystemActorOf(
```

scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala, line 90 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: isTerminated
Enclosing Method: DistributedData()
File: scala/akka/cluster/ddata/typed/scaladsl/DistributedData.scala:90
Taint Flags:

```
87 * @see [[DistributedData.withReplicatorMessageAdapter]]
88 */
89 val replicator: ActorRef[Replicator.Command] =
90 if (isTerminated) {
91 val log = LoggerFactory.getLogger(getClass)
92 if (Cluster(classicSystem).isTerminated)
93 log.warn("Replicator points to dead letters, because Cluster is terminated.")
```

Package: akka.cluster.typed.internal

scala/akka/cluster/typed/internal/AdaptedClusterImpl.scala, line 159 (Code Correctness: Constructor Invokes Overridable Function) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: classicCluster
Enclosing Method: AdapterClusterImpl()
File: scala/akka/cluster/typed/internal/AdaptedClusterImpl.scala:159
Taint Flags:

```
156 override def state: ClusterEvent.CurrentClusterState = classicCluster.state
157
158 // must not be lazy as it also updates the cached selfMember
```

Code Correctness: Constructor Invokes Overridable Function	Low

Package: akka.cluster.typed.internal

scala/akka/cluster/typed/internal/AdaptedClusterImpl.scala, line 159 (Code Correctness: Constructor Invokes Overridable Function)	Low

```
159 override val subscriptions: ActorRef[ClusterStateSubscription] =
160 system.internalSystemActorOf(
161 // resume supervision: has state that shouldn't be lost in case of failure
162 Behaviors.supervise(subscriptionsBehavior(classicCluster)).onFailure(SupervisorStrategy.resume),
```

scala/akka/cluster/typed/internal/AdaptedClusterImpl.scala, line 159 (Code Correctness: Constructor Invokes Overridable Function)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: FunctionCall: akka$cluster$typed$internal$AdapterClusterImpl$$subscriptionsBehavior
Enclosing Method: AdapterClusterImpl()
File: scala/akka/cluster/typed/internal/AdaptedClusterImpl.scala:159
Taint Flags:

```
156 override def state: ClusterEvent.CurrentClusterState = classicCluster.state
157
158 // must not be lazy as it also updates the cached selfMember
159 override val subscriptions: ActorRef[ClusterStateSubscription] =
160 system.internalSystemActorOf(
161 // resume supervision: has state that shouldn't be lost in case of failure
162 Behaviors.supervise(subscriptionsBehavior(classicCluster)).onFailure(SupervisorStrategy.resume),
```

Code Correctness: Erroneous String Compare (3 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");
  }
```

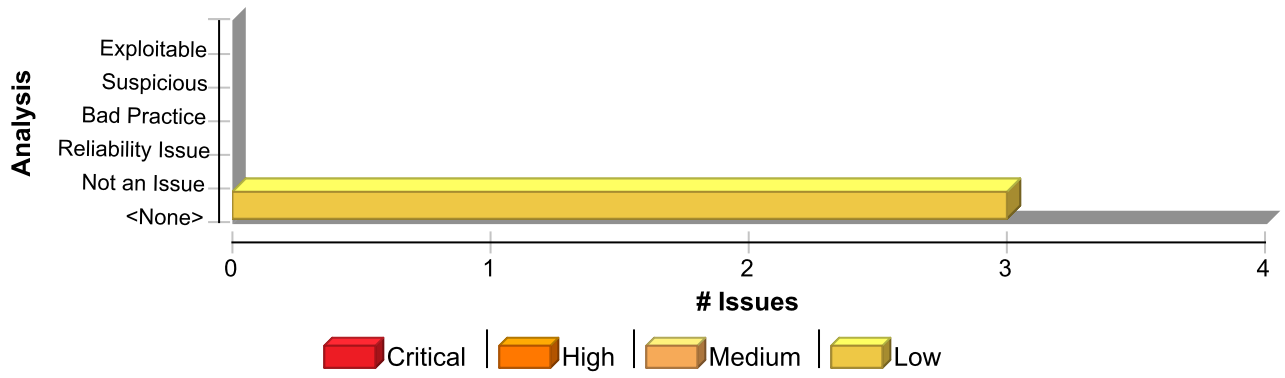
The `==` and `!=` operators will only behave as expected when they are used to compare strings contained in objects that are equal. The most common way for this to occur is for the strings to be interned, whereby the strings are added to a pool of objects maintained by the `String` class. Once a string 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	3	0	0	3
Total	**3**	**0**	**0**	**3**

Code Correctness: Erroneous String Compare — Low

Package: akka.cluster.typed.internal.receptionist

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala, line 30 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: apply()
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala:30
Taint Flags:

```
27 val writeTimeout = 5.seconds // the timeout is not important
28 val writeConsistency = {
29 val key = "write-consistency"
30 toRootLowerCase(config.getString(key)) match {
31 case "local" => Replicator.WriteLocal
32 case "majority" => Replicator.WriteMajority(writeTimeout)
33 case "all" => Replicator.WriteAll(writeTimeout)
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala, line 30 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Operation
Enclosing Method: apply()
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala:30
Taint Flags:

```
27 val writeTimeout = 5.seconds // the timeout is not important
28 val writeConsistency = {
29 val key = "write-consistency"
30 toRootLowerCase(config.getString(key)) match {
31 case "local" => Replicator.WriteLocal
32 case "majority" => Replicator.WriteMajority(writeTimeout)
33 case "all" => Replicator.WriteAll(writeTimeout)
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala, line 30 (Code Correctness: Erroneous String Compare) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Code Correctness: Erroneous String Compare	Low
Package: akka.cluster.typed.internal.receptionist	
scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala, line 30 (Code Correctness: Erroneous String Compare)	**Low**

Sink Details

Sink: Operation
Enclosing Method: apply()
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala:30
Taint Flags:

```
27 val writeTimeout = 5.seconds // the timeout is not important
28 val writeConsistency = {
29 val key = "write-consistency"
30 toRootLowerCase(config.getString(key)) match {
31 case "local" => Replicator.WriteLocal
32 case "majority" => Replicator.WriteMajority(writeTimeout)
33 case "all" => Replicator.WriteAll(writeTimeout)
```

Code Correctness: Non-Static Inner Class Implements Serializable (10 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

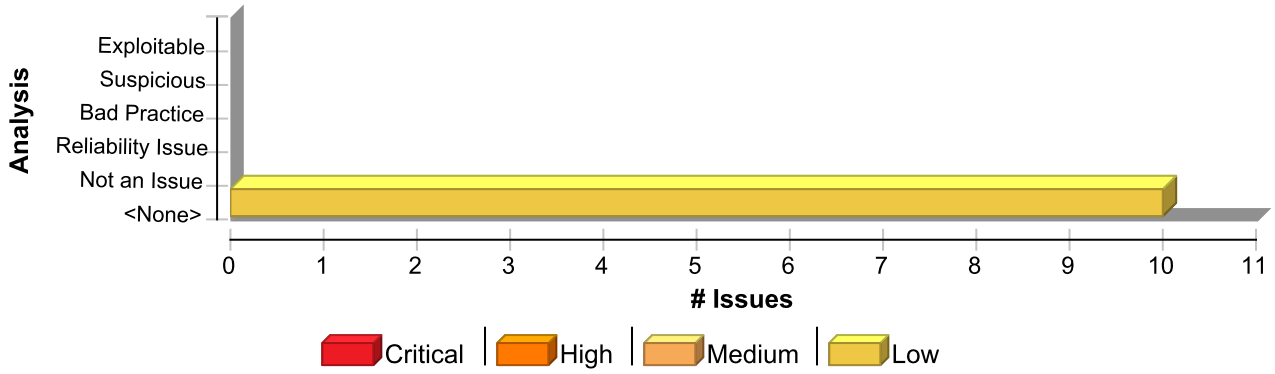
When using inner classes, they should not be serialized, or they should be changed to static-nested classes, since these do not have the drawbacks that non-static inner classes have when serialized. When a nested class is static it inherently has no association with instance variables (including those 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	10	0	0	10
Total	**10**	**0**	**0**	**10**

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

Package: akka.cluster.typed.internal

scala/akka/cluster/typed/internal/AdaptedClusterImpl.scala, line 29 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: AdapterClusterImpl$Removed
File: scala/akka/cluster/typed/internal/AdaptedClusterImpl.scala:29
Taint Flags:

```
26 private sealed trait SeenState
27 private case object BeforeUp extends SeenState
28 private case object Up extends SeenState
29 private case class Removed(previousStatus: MemberStatus) extends SeenState
30
31 private def subscriptionsBehavior(adaptedCluster: akka.cluster.Cluster): Behavior[ClusterStateSubscription] =
32 Behaviors.setup[ClusterStateSubscription] { ctx =>
```

Package: akka.cluster.typed.internal.receptionist

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 66 (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.cluster.typed.internal.receptionist

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 66 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Sink: Class: ClusterReceptionist$NodeUnreachable
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala:66
Taint Flags:

```
63 extends InternalCommand
64 private final case class NodeAdded(addresses: UniqueAddress) extends InternalCommand
65 private final case class NodeRemoved(addresses: UniqueAddress) extends InternalCommand
66 private final case class NodeUnreachable(addresses: UniqueAddress) extends InternalCommand
67 private final case class NodeReachable(addresses: UniqueAddress) extends InternalCommand
68 private final case class ChangeFromReplicator(key: DDataKey, value: ORMultiMap[ServiceKey[_], Entry])
69 extends InternalCommand
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 68 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ClusterReceptionist$ChangeFromReplicator
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala:68
Taint Flags:

```
65 private final case class NodeRemoved(addresses: UniqueAddress) extends InternalCommand
66 private final case class NodeUnreachable(addresses: UniqueAddress) extends InternalCommand
67 private final case class NodeReachable(addresses: UniqueAddress) extends InternalCommand
68 private final case class ChangeFromReplicator(key: DDataKey, value: ORMultiMap[ServiceKey[_], Entry])
69 extends InternalCommand
70 private case object RemoveTick extends InternalCommand
71 private case object PruneTombstonesTick extends InternalCommand
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 80 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ClusterReceptionist$State
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala:80
Taint Flags:

```
77 * for a key.
78 * @param subscriptions Locally subscriptions, not replicated
```

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

Package: akka.cluster.typed.internal.receptionist

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 80 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

```
79 */
80 final case class State(
81 registry: ShardedServiceRegistry,
82 servicesPerActor: Map[ActorRef[_], Set[AbstractServiceKey]],
83 tombstones: Map[ActorRef[_], Set[(AbstractServiceKey, Deadline)]],
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.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: ClusterReceptionist$Entry
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala:51
Taint Flags:

```
48
49 // values contain system uid to make it possible to discern actors at the same
50 // path in different incarnations of a cluster node
51 final case class Entry(ref: ActorRef[_], systemUid: Long)(val createdTimestamp: Long) {
52 def uniqueAddress(selfAddress: Address): UniqueAddress =
53 if (ref.path.address.hasLocalScope) UniqueAddress(selfAddress, systemUid)
54 else UniqueAddress(ref.path.address, systemUid)
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 67 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ClusterReceptionist$NodeReachable
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala:67
Taint Flags:

```
64 private final case class NodeAdded(addresses: UniqueAddress) extends InternalCommand
65 private final case class NodeRemoved(addresses: UniqueAddress) extends InternalCommand
66 private final case class NodeUnreachable(addresses: UniqueAddress) extends InternalCommand
67 private final case class NodeReachable(addresses: UniqueAddress) extends InternalCommand
68 private final case class ChangeFromReplicator(key: DDataKey, value: ORMultiMap[ServiceKey[_], Entry])
69 extends InternalCommand
70 private case object RemoveTick extends InternalCommand
```

Code Correctness: Non-Static Inner Class Implements Serializable	Low

Package: akka.cluster.typed.internal.receptionist

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 67 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 62 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ClusterReceptionist$SubscriberTerminated
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala:62
Taint Flags:

```
59
60 private sealed trait InternalCommand extends Command
61 private final case class LocalServiceActorTerminated[T](ref: ActorRef[T]) extends InternalCommand
62 private final case class SubscriberTerminated[T](ref: ActorRef[ReceptionistMessages.Listing[T]])
63 extends InternalCommand
64 private final case class NodeAdded(addresses: UniqueAddress) extends InternalCommand
65 private final case class NodeRemoved(addresses: UniqueAddress) extends InternalCommand
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 61 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ClusterReceptionist$LocalServiceActorTerminated
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala:61
Taint Flags:

```
58 }
59
60 private sealed trait InternalCommand extends Command
61 private final case class LocalServiceActorTerminated[T](ref: ActorRef[T]) extends InternalCommand
62 private final case class SubscriberTerminated[T](ref: ActorRef[ReceptionistMessages.Listing[T]])
63 extends InternalCommand
64 private final case class NodeAdded(addresses: UniqueAddress) extends InternalCommand
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 64 (Code Correctness: Non-Static Inner Class Implements Serializable)	Low

Issue Details

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

Package: akka.cluster.typed.internal.receptionist

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 64 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ClusterReceptionist$NodeAdded
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala:64
Taint Flags:

```
61 private final case class LocalServiceActorTerminated[T](ref: ActorRef[T]) extends InternalCommand
62 private final case class SubscriberTerminated[T](ref: ActorRef[ReceptionistMessages.Listing[T]])
63 extends InternalCommand
64 private final case class NodeAdded(addresses: UniqueAddress) extends InternalCommand
65 private final case class NodeRemoved(addresses: UniqueAddress) extends InternalCommand
66 private final case class NodeUnreachable(addresses: UniqueAddress) extends InternalCommand
67 private final case class NodeReachable(addresses: UniqueAddress) extends InternalCommand
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala, line 65 (Code Correctness: Non-Static Inner Class Implements Serializable) — Low

Issue Details

Kingdom: Code Quality
Scan Engine: SCA (Structural)

Sink Details

Sink: Class: ClusterReceptionist$NodeRemoved
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionist.scala:65
Taint Flags:

```
62 private final case class SubscriberTerminated[T](ref: ActorRef[ReceptionistMessages.Listing[T]])
63 extends InternalCommand
64 private final case class NodeAdded(addresses: UniqueAddress) extends InternalCommand
65 private final case class NodeRemoved(addresses: UniqueAddress) extends InternalCommand
66 private final case class NodeUnreachable(addresses: UniqueAddress) extends InternalCommand
67 private final case class NodeReachable(addresses: UniqueAddress) extends InternalCommand
68 private final case class ChangeFromReplicator(key: DDataKey, value: ORMultiMap[ServiceKey[_], Entry])
```

Key Management: Hardcoded Encryption Key (2 issues)

Abstract

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

Explanation

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

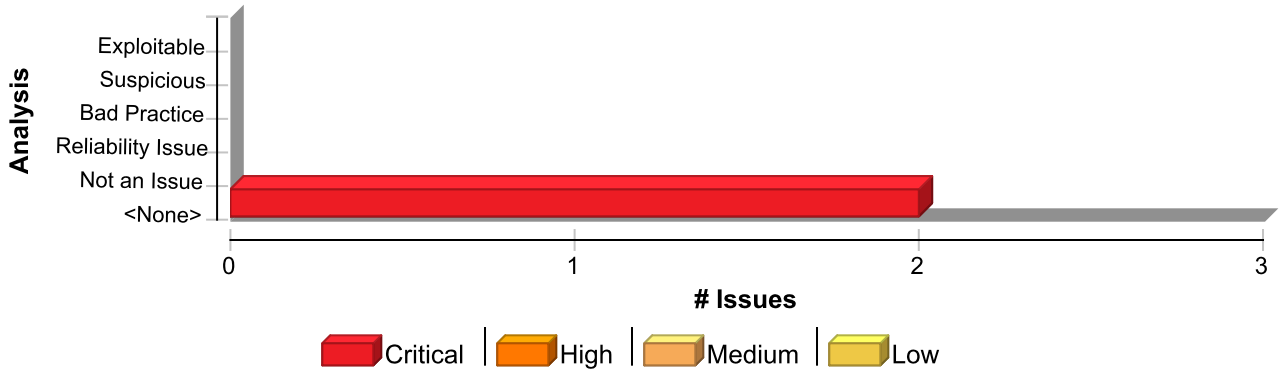
```
...
private static final String encryptionKey = "lakdsljkalkjlksdfkl";
byte[] keyBytes = encryptionKey.getBytes();
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher encryptCipher = Cipher.getInstance("AES");
encryptCipher.init(Cipher.ENCRYPT_MODE, key);
...
```

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

Recommendation

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

Issue Summary



Engine Breakdown

	SCA	WebInspect	SecurityScope	Total
Key Management: Hardcoded Encryption Key	2	0	0	2
Total	**2**	**0**	**0**	**2**

Key Management: Hardcoded Encryption Key — Critical

Package: akka.cluster.typed.internal.receptionist

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala, line 29 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: key
Enclosing Method: apply()
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala:29
Taint Flags:

```
26 def apply(config: Config): ClusterReceptionistSettings = {
27 val writeTimeout = 5.seconds // the timeout is not important
28 val writeConsistency = {
29 val key = "write-consistency"
30 toRootLowerCase(config.getString(key)) match {
31 case "local" => Replicator.WriteLocal
32 case "majority" => Replicator.WriteMajority(writeTimeout)
```

scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala, line 29 (Key Management: Hardcoded Encryption Key) — Critical

Issue Details

Kingdom: Security Features
Scan Engine: SCA (Structural)

Sink Details

Sink: VariableAccess: key
Enclosing Method: apply()
File: scala/akka/cluster/typed/internal/receptionist/ClusterReceptionistSettings.scala:29
Taint Flags:

```
26 def apply(config: Config): ClusterReceptionistSettings = {
27 val writeTimeout = 5.seconds // the timeout is not important
28 val writeConsistency = {
29 val key = "write-consistency"
30 toRootLowerCase(config.getString(key)) match {
31 case "local" => Replicator.WriteLocal
32 case "majority" => Replicator.WriteMajority(writeTimeout)
```