

apache / incubator-seata Public

<> Code Issues 794 Pull requests 82 Discussions Actions Projects

# Commit 20cd962



funky-eyes authored on Jul 27, 2024 Verified

security: fix REC security vulnerability during Raft deserialization (#6702)

2.x (#6702) · v2.6.0 ... v2.2.0

1 parent [b34cfff](#) commit 20cd962

7 files changed +95 -10 lines changed

[↑ Top](#)

Filter files...

- changes
    - en-us
      - 2.x.md
    - zh-cn
      - 2.x.md
  - common/src/main
    - java/org/apache/seata/common/exception
      - ErrorCode.java
    - resources/error
      - ErrorCode\_en.properties
  - server/src
    - main/java/org/apache/seata/server/cluster/raft/sync
      - RaftSyncMessageSerializer.java
    - test/java/org/apache/seata/server/raft
      - RaftSyncMessageTest.java

TestSecurity.java

7 files changed +95 -10 lines changed

Search within code



```

changes/en-us/2.x.md
@@ -36,7 +36,7 @@ Add changes here for all PR submitted to the 2.x branch.
36 36   ### refactor:
37 37
38 38   ### security:
39 39   -
39 39   + - [[#6702](https://github.com/apache/incubator-seata/pull/6702)] fix REC
security vulnerability during Raft deserialization
40 40
41 41   ### test:
42 42   - [[#6608](https://github.com/apache/incubator-seata/pull/6608)] add unit test
for sql-parser-core

```

```

changes/zh-cn/2.x.md
@@ -38,6 +38,7 @@
38 38
39 39
40 40   ### security:
41 41   + - [[#6702](https://github.com/apache/incubator-seata/pull/6702)] 修复Raft反序列化
时存在RCE安全漏洞
41 42
42 43   ### test:
43 44   - [[#6608](https://github.com/apache/incubator-seata/pull/6608)] 添加sql-parser-
core模块测试用例

```

```

...pache/seata/common/exception/ErrorCode.java
@@ -24,11 +24,16 @@ public enum ErrorCode {
24 24     /**
25 25     * 0001 ~ 0099 Configuration related errors
26 26     */
27 27     - ERR_CONFIG(ErrorType.Config, 0001);
27 27     + ERR_CONFIG(ErrorType.Config, 0001),

```

```

28  +
28  29      /**
29  -      * The error code of the transaction exception.
30  +      * 0100 ~ 0199 Security related errors
30  31      */
32  +      ERR_DESERIALIZATION_SECURITY(ErrorType.Security, 0156);
31  33
34  +      /**
35  +      * The error code of the transaction exception.
36  +      */
32  37      private int code;
33  38      private ErrorType type;
34  39
    ↓
    ↑      @@ -77,6 +82,10 @@ enum ErrorType {
77  82          * Network error type.
78  83          */
79  84          Network,
85  +      /**
86  +      * Security related error type.
87  +      */
88  +      Security,
80  89          /**
81  90          * Tm error type.
82  91          */
    ↓

```

```

  ...ain/resources/error/ErrorCode_en.properties
    ↑      @@ -16,4 +16,5 @@
16  16      #
17  17      ERR_PREFIX=ERR-CODE: [Seata-{code}][{key}]
18  18      ERR_POSTFIX=More: [https://seata.apache.org/docs/next/overview/faq#{code}]
19  -      ERR_CONFIG=config error, {0}
    ⊖
19  +      ERR_CONFIG=config error, {0}
20  +      ERR_DESERIALIZATION_SECURITY=deserialization security error, {0}
    ⊖

```

```

  ...er/raft/sync/RaftSyncMessageSerializer.java

```

	↑	@@ -21,7 +21,12 @@
21	21	import java.io.IOException;
22	22	import java.io.ObjectInputStream;
23	23	import java.io.ObjectOutputStream;
	24	+ import java.io.ObjectStreamClass;
	25	+ import java.util.ArrayList;
	26	+ import java.util.List;
24	27	import java.util.Optional;
	28	+ import org.apache.seata.common.exception.ErrorCode;
	29	+ import org.apache.seata.common.exception.SeataRuntimeException;
25	30	import org.apache.seata.common.loader.EnhancedServiceLoader;
26	31	import org.apache.seata.core.compressor.CompressorFactory;
27	32	import org.apache.seata.core.serializer.Serializer;
	↕	@@ -36,6 +41,14 @@ public class RaftSyncMessageSerializer {
36	41	
37	42	private static final Logger LOGGER =
		LoggerFactory.getLogger(RaftSyncMessageSerializer.class);
38	43	
	44	+ private static final List<String> PERMITS = new ArrayList<>();
	45	+
	46	+ static {
	47	+ PERMITS.add(RaftSyncMessage.class.getName());
	48	+ PERMITS.add(io.seata.server.cluster.raft.sync.msg.RaftSyncMessage.class.getName());
	49	+ PERMITS.add("[B");
	50	+ }
	51	+
39	52	public static byte[] encode(RaftSyncMessage raftSyncMessage) throws
		IOException {
40	53	try (ByteArrayOutputStream bos = new ByteArrayOutputStream();
41	54	ObjectOutputStream oos = new ObjectOutputStream(bos)) {
	↕	@@ -62,12 +75,22 @@ public static byte[]
		encode(io.seata.server.cluster.raft.sync.msg.RaftSyncMessag
62	75	
63	76	public static RaftSyncMessage decode(byte[] raftSyncMsgByte) {
64	77	try (ByteArrayInputStream bin = new
		ByteArrayInputStream(raftSyncMsgByte);
65	-	ObjectInputStream ois = new ObjectInputStream(bin)) {
	78	+ ObjectInputStream ois = new ObjectInputStream(bin) {

```

79 +         @Override
80 +         protected Class<?> resolveClass(ObjectStreamClass desc) throws
IOException, ClassNotFoundException {
81 +             if (!PERMITS.contains(desc.getName())) {
82 +                 throw new
SeataRuntimeException(ErrorCode.ERR_DESERIALIZATION_SECURITY,
83 +                     "Failed to deserialize object: " + desc.getName() +
" is not permitted");
84 +             }
85 +
86 +             return super.resolveClass(desc);
87 +         }
88 +     }) {
66 89         Object object = ois.readObject();
67 90         RaftSyncMessage raftSyncMessage;
68 91         if (object instanceof
io.seata.server.cluster.raft.sync.msg.RaftSyncMessage) {
69 92             io.seata.server.cluster.raft.sync.msg.RaftSyncMessage
oldRaftSyncMessage =
70 -
(io.seata.server.cluster.raft.sync.msg.RaftSyncMessage)object;
93 +
(io.seata.server.cluster.raft.sync.msg.RaftSyncMessage)object;
71 94             raftSyncMessage = new RaftSyncMessage();
72 95             raftSyncMessage.setCodec(oldRaftSyncMessage.getCodec());
73 96
raftSyncMessage.setCompressor(oldRaftSyncMessage.getCompressor());
@@ -77,13 +100,16 @@ public static RaftSyncMessage decode(byte[]
raftSyncMsgByte) {
77 100             raftSyncMessage = (RaftSyncMessage)object;
78 101         }
79 102         Serializer serializer =
EnhancedServiceLoader.load(Serializer.class,
80 -
SerializerType.getByCode(raftSyncMessage.getCodec()).name());
103 +
SerializerType.getByCode(raftSyncMessage.getCodec()).name());
81 104         Optional.ofNullable(raftSyncMessage.getBody())
82 -
.isPresent(value ->
raftSyncMessage.setBody(serializer.deserialize(CompressorFactory

```

```

83      -
      .getCompressor(raftSyncMessage.getCompressor()).decompress((byte[])raftSyncMess
      age.getBody()))));
105     +
      .ifPresent(value ->
      raftSyncMessage.setBody(serializer.deserialize(CompressorFactory
106     +
      .getCompressor(raftSyncMessage.getCompressor()).decompress((byte[])raftSyncMess
      age.getBody()))));
84     107
      return raftSyncMessage;
85     -
      } catch (ClassNotFoundException | IOException e) {
108    +
      } catch (Exception e) {
86     109
      LOGGER.info("Failed to read raft synchronization log: {}",
      e.getMessage(), e);
110    +
      if (e instanceof SeataRuntimeException) {
111    +
      throw (SeataRuntimeException)e;
112    +
      }
87     113
      throw new RuntimeException(e);
88     114
      }
89     115
      }

```

```

.../seata/server/raft/RaftSyncMessageTest.java
@@ -16,12 +16,15 @@
16 16    */
17 17    package org.apache.seata.server.raft;
18 18
19 19    + import java.io.ByteArrayOutputStream;
19 20    import java.io.IOException;
21 21    + import java.io.ObjectOutputStream;
20 22    import java.util.ArrayList;
21 23    import java.util.HashMap;
22 24    import java.util.List;
23 25    import java.util.Map;
24 26
27 27    + import org.apache.seata.common.exception.SeataRuntimeException;
25 28    import org.apache.seata.common.metadata.ClusterRole;
26 29    import org.apache.seata.common.metadata.Node;
27 30    import org.apache.seata.core.exception.TransactionException;
@@ -62,6 +65,19 @@ public static void setUp(ApplicationContext context){

```

```

62 65      public static void destroy(){
63 66          SessionHolder.destroy();
64 67      }

68 +
69 +      @Test
70 +      public void testSecurityMsgSerialize() throws IOException {
71 +          TestSecurity testSecurity = new TestSecurity();
72 +          byte[] bytes;
73 +          try (ByteArrayOutputStream bos = new ByteArrayOutputStream();
74 +              ObjectOutputStream oos = new ObjectOutputStream(bos)) {
75 +              oos.writeObject(testSecurity);
76 +              bytes = bos.toByteArray();
77 +          }
78 +          Assertions.assertThrows(SeataRuntimeException.class, () -
>RaftSyncMessageSerializer.decode(bytes));
79 +      }
80 +

65 81      @Test
66 82      public void testMsgSerialize() throws IOException {
67 83          RaftSyncMessage raftSyncMessage = new RaftSyncMessage();

```



▼ .../apache/seata/server/raft/TestSecurity.java



```

...    @@ -0,0 +1,32 @@
1  + /*
2  +  * Licensed to the Apache Software Foundation (ASF) under one or more
3  +  * contributor license agreements. See the NOTICE file distributed with
4  +  * this work for additional information regarding copyright ownership.
5  +  * The ASF licenses this file to You under the Apache License, Version 2.0
6  +  * (the "License"); you may not use this file except in compliance with
7  +  * the License. You may obtain a copy of the License at
8  +  *
9  +  *     http://www.apache.org/licenses/LICENSE-2.0
10 +  *
11 +  * Unless required by applicable law or agreed to in writing, software
12 +  * distributed under the License is distributed on an "AS IS" BASIS,
13 +  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
14 +  * See the License for the specific language governing permissions and
15 +  * limitations under the License.
16 +  */

```

```
17 + package org.apache.seata.server.raft;
18 +
19 + public class TestSecurity implements java.io.Serializable {
20 +
21 +     private static final long serialVersionUID = 543214259201495900L;
22 +
23 +     String a = "test";
24 +
25 +     public String getA() {
26 +         return a;
27 +     }
28 +
29 +     public void setA(String a) {
30 +         this.a = a;
31 +     }
32 + }
```

## Comments 0



Please [sign in](#) to comment.