Reproducing The ProxyShell Pwn2Own Exploit

Peterjson · Follow
7 min read · Aug 6, 2021

INTRO

I and Jang recently successfully reproduced the ProxyShell Pwn2Own Exploit of Orange Tsai 🍊. Firstly, I just want to tell that I respect your hard work and the contribution of you to cybersecurity which inspired me many years ago. Now I want to summary the progress when we reproduce this Exploit chain as a write-up for our-self. When ZDI release the advisories about these bug, I decided to analysis this chain for learning purpose. We can almost finished the chain before the BlackHat US talk's of Orange and then we found an missing piece when Orange's talk finished. So this write-up as an progress of 1-day analysis, we diff the patch, and solved the puzzle step by step. For those who never seen Orange's talk, you can check it here.

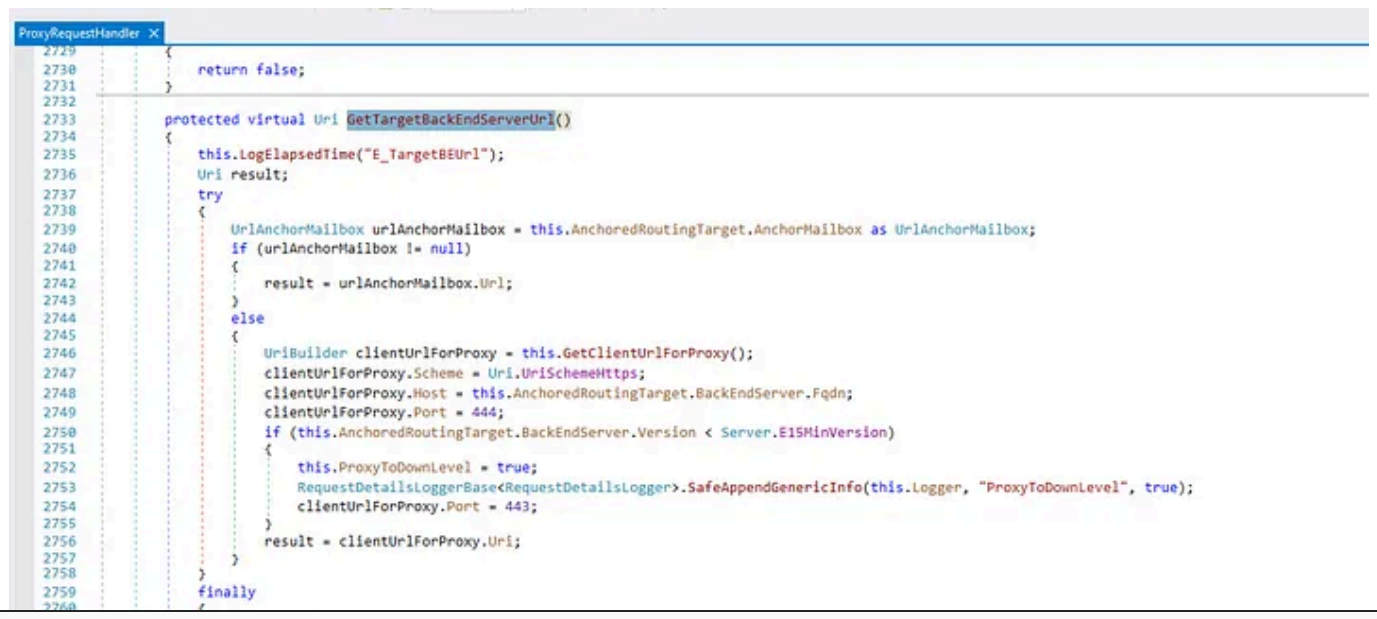
For each front-end endpoint like */ecp/*, */owa/*, */autodiscover/*, */powershell/* and so

M

from ProxyLogon analysis

ProxyLogon entry

From ProxyLogon, we know that we can set *AnchoredRoutingTarget* variable from "*X-BEResource*", then Exchange when calculate the target backend URL to request internal we can reach internal endpoint we overwrite it and we have SSRF



Microsoft.Exchange.HttpProxy.AutodiscoverProxyRequestHandler

```
AutodiscoverProxyRequestHandler
  => implement EwsAutodiscoverProxyRequestHandler
   => implement BEServerCookieProxyRequestHandler
    => implement ProxyRequestHandler
```

And “/autodiscover” also allow for unauthenticated

```
ProxyRequestHandler
2737            try
2738            {
2739                UrlAnchorMailbox urlAnchorMailbox = this.AnchoredRoutingTarget.AnchorMailbox as UrlAnchorMailbox;
2740                if (urlAnchorMailbox != null)
2741                {
2742                    result = urlAnchorMailbox.Url;
2743                }
2744                else
2745                {
2746                    UriBuilder clientUrlForProxy = this.GetClientUrlForProxy();
2747                    clientUrlForProxy.Scheme = Uri.UriSchemeHttps;
2748                    clientUrlForProxy.Host = this.AnchoredRoutingTarget.BackEndServer.Fqdn;
2749                    clientUrlForProxy.Port = 444;
2750                    if (this.AnchoredRoutingTarget.BackEndServer.Version < Server.E15MinVersion)
2751                    {
2752                        this.ProxyToDownLevel = true;
2753                        RequestDetailsLoggerBase<RequestDetailsLogger>.SafeAppendGenericInfo(this.Logger, "ProxyToDownLevel", true);
2754                        clientUrlForProxy.Port = 443;
2755                    }
2756                    result = clientUrlForProxy.Uri;
2757                }
2758            }
2759            finally
2760            {
2761                this.LogElapsedTime("L_TargetBEUrl");
2762            }
2763            return result;
2764        }
2765
```

ProxyRequestHandler.GetTargetBackEndServerUrl()

After ProxyLogon patch, there's a check for *AnchoredRoutingTarget* variable, so we somehow can successfully change it again like ProxyLogon, we will got 503 , don't know why? check here

ProxyRequestHandler.GetTargetBackEndServerUrl() will return the URI after finish calculate, we cannot abuse *AnchoredRoutingTarget* anymore, how about GetClientUrlForProxy() ? Then control our URI and send into backend, sound interesting

How can we set explicitLogonAddress variable from our request?

Notice that, Params variable contains parameters from query string, form parameter, cookies, ...

We need to pass some conditions

1. We want to reach the if statement so *IsAutodiscoverV2Request()* must return ***False*** and *IsAutodiscoverV2PreviewRequest()* return ***False*** also

So it is /autodiscover/autodiscover.json + dummy string

2.

So it is "/autodiscover/autodiscover.json?a=dummy@dummy.pw" (in order to help us can reach the if statement which will return ***False*** and ***remove explicitLogon***) and then we set this value into Email Cookie with the same value

3. When preparing request to send to backend internal, Exchange will generate Kerberos auth header and attach into Authorization header. This is why we can reach some other endpoint without any authentication

PrepareServerRequest()

Chaining into together we have an pre-auth SSRF

We don't have permission on this endpoint :(

Always remember one think, you should understand on what you are looking while doing 1-day anlysis. IIS has some modules on each web application, they are excuted before the actual handler executed. You can imagine they're like "filter" mechanism on Java web apps.

Powershell-Proxy IIS modules

We need to look at each module to see what we have missed. On BackendRehydrationModule when process the request, this module cannot get CommonAccessToken (from Exchange SSRF) there will be an exception and we cannot go through.

BackendRehydrationModule.ProcessRequest()

So how can we set the header “X-CommonAccessToken” because we cannot make Exchange copy it to SSRF request and send to “/powershell”

some blacklist cookies Exchange won’t copy to internal

Before BackendRehydrationModule executed, there’s RemotePowershellBackendCmdletProxyModule

Basically, when the SSRF doesn't contain Header "X-CommonAccessToken" Ex

ou

BackendRehydrationModule we will survive from the Exception. But how can we create a valid CommonAccessToken or maybe high privilege CommonAccessToken? We need to reverse the structure of CommonAccessToken

deserialize "X-Rps-CAT" into CommonAccessToken

```
V + version + T + type + C + compress + data
if compress => decompress then if type is Windows
```

This is pseudocode I make for CommonAccessToken, if the token type is "Windows", Exchange continue deserialise our data

finally foward into 4443. With this setup, we can capture a "sample"

Co

How can you get a valid user SUID without exist user on Exchange? When deploying Exchange, there are some "always exist" mailbox such as

https://docs.microsoft.com/en-us/exchange/architecture/mailbox-servers/recreate-arbitration-mailboxes?

From MS docs, this cmdlet can export the mailbox into arbitrary location

Th

RC

https://docs.microsoft.com/en-us/powershell/module/exchange/new-mailboxexportrequest?view=exchange-ps

We can confirm it again, because the patch only allow some specific extension

But how can we control the data in the mailbox and make it into shell after the file was exported? This is what we got stuck for a long time until Orange's talk appear.

https://docs.microsoft.com/en-us/openspecs/office_file_formats/ms-pst/5faf4800-645d-49d1-9457-2ac40eb467bd

But how can we put our shell into the mailbox and then export it as our shell?

EWS will save us, EWS (**/ews/exchange.asmx**) is a service based on SOAP which help us can create mail, event, meeting, …

We can create an email saved in “drafts” for any user via SOAP header “SerializedSecurityContext”- this called EWS Impersonation . Then injecting our “encoded” shell as an attachment.

Chaining all together

No
implement an WinRM protocol for the Pre-auth SSRF to communicate with "/powershell" endpoint. I leave this as an lesson for reader and hopefully you should reproduce this bug by yourself because it help you learn many things.

For myself, I use pypsrp then collect the data while it processing and plug it into our SSRF. To understand more about WinRM you can check this awesome blog

Or you can do the same with Orange's way, implement his own proxy to communicate with WinRM

Our demonstration:

https://www.youtube.com/watch?v=LbIYPFrltdA

Written by Peterjson

Follow

418 Followers

Nub-boi

More from Peterjson

Peterjson

Miracle - One Vulnerability To Rule Them All

Introduction

Jun 23, 2022 151 3

Peterjson in tradahacking

[RMI] Study Note And Some Study Case

Hi ! Lâu rồi mình cũng không viết blog hay là write-up về CTF nữa, mà lần này mình muốn...

Feb 23, 2020 4

RED TEAM

Malware Development Part 8 : Reverse Shell Via Dll Hijacking

"From DLL to Shell: A Step-by-Step Guide to Reverse Shell via DLL Hijacking"

Jun 22 147

jayjonah.eth

$150,000 Evmos Vulnerability Through Reading Documentation

Life as a Web3 security researcher often consists of deep diving into technical subjec...

5d ago 64 1

See more recommendations

Help Status About Careers Press Blog Privacy Terms Text to speech Teams