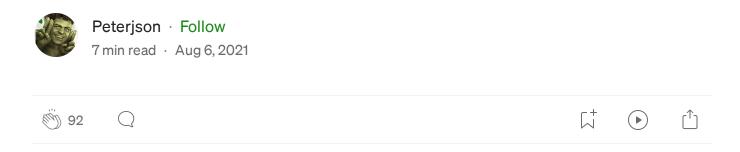


# Reproducing The ProxyShell Pwn2Own Exploit

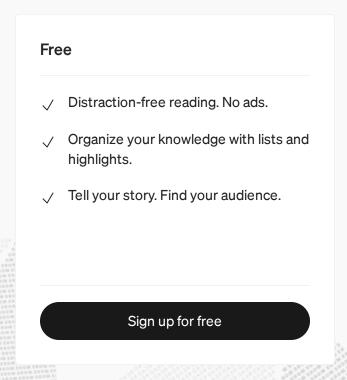


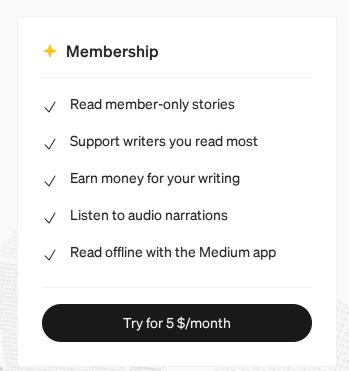
#### **INTRO**

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.

## **Medium**

Sign up to discover human stories that deepen your understanding of the world.





X

```
For each front and and point like /ecn/ /owa/ /autodiscover/ /nowershell/ and so To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy, including cookie policy.

M
```

from ProxyLogon analysis

```
ProxyRequestHandler @020000AE

DExposed By

Lettension Methods

Used By

DExtension Methods

DExposed By

DExtension Methods

DExposed By

DExtension Methods

DExtension Methods

DExtension Methods

DExtension Methods

DInstantiated By

DExtension Methods

DInstantiated By

DExtension Methods

DInstantiated By

DExtension Methods

DINGROUPE Control of the Microsoft Exchange HttpProxy. E4eProxyRequestHandler. IsRequestBoundToBEServer(HttpRequest): bool @060004E7

DINGROUPE CONTROL OF THE MICROSOft Exchange HttpProxy. E4eProxyRequestHandler. ResolveAnchorMailbox (): AnchorMailbox @060004BB

DINGROUPE Control of the Microsoft Exchange HttpProxy. OwaEcpProxyRequestHandler CanHandler (Http Request): bool @0600059A

DINGROUPE Control of the Microsoft Exchange HttpProxy. OwaEcpProxyRequestHandler CanHandler (Http Request): bool @060005BB

DINGROUPE Control of the Microsoft Exchange HttpProxy. OwaEcpProxyRequestHandler CanHandler (Http Request): IHttpHandler @060005F4

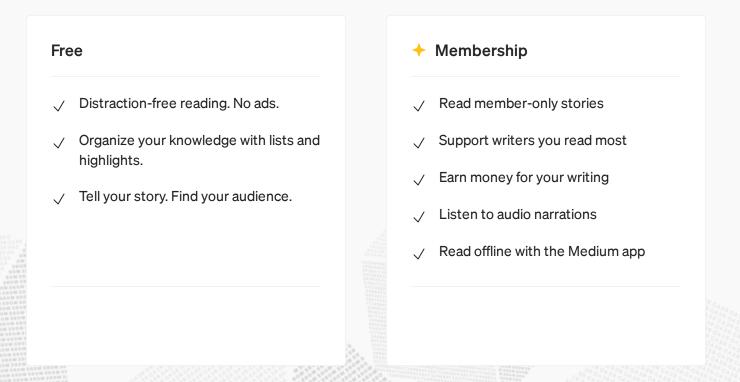
DINGROUPE Control of the Microsoft Exchange HttpProxy. ProxyModule. SelectHandler ForUnauthenticatedRequest (HttpContext): IHttpHandler @060005F5
```

**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

```
2733
2734
2735
                  protected virtual Uri GetTargetBackEndServerUrl()
                       this.LogElapsedTime("E_TargetBEUrl");
2736
2739
                            UrlAnchorMailbox urlAnchorMailbox = this.AnchoredRoutingTarget.AnchorMailbox as UrlAnchorMailbox;
2740
2741
2742
                            if (urlAnchorMailbox != null)
                                 result = urlAnchorMailbox.Url;
2743
2744
2745
2746
                                 UriBuilder clientUrlForProxy - this.GetClientUrlForProxy();
                                 clientUrlForProxy.Scheme = Uri.UriSchemeHttps;
clientUrlForProxy.Host = this.AnchoredRoutingTarget.BackEndServer.Fqdn;
2747
2748
                                 clientUrlForProxy.Port = 444;
if (this.AnchoredRoutingTarget.BackEndServer.Version < Server.E1SMinVersion)</pre>
2749
2752
                                      this.ProxyToDownLevel = true;
2753
2754
2755
                                     RequestDetailsLoggerBase<Reque
clientUrlForProxy.Port = 443;
                                                                         puestDetailsLogger>.SafeAppendGenericInfo(this.Logger, "ProxyToDownLevel", true);
2756
2757
2758
                                 result = clientUrlForProxy.Uri;
```

# Medium



```
ProxyRequestHandler @020000AE
                                To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy,
                                including cookie policy.
                                D Extension Methods
                                ▲ D Used By
                                         ComplianceServiceProxyRequestHandler @0200008E

EwsAutodiscoverProxyRequestHandler @02000093
                                          MailboxDeliveryProxyRequestHandler @0200009B

    MapiProxyRequestHandler @0200009C
    MicroServiceProxyRequestHandler @020000A3

                                            ▶ ♠ Microsoft.Exchange.HttpProxy.AnonymousCalendarProxyRequestHandler.AnonymousCalendarProxyRequestHandler(): void @0600049€
                                                                                                                                                                                    ousCalendarProxyRequestHandler.ResolveAnchorMailbox(): AnchorMailbox @0600049A
                                          ▶ ♥. Microsoft.Exchange.HttpProxy.AutodiscoverProxyRequestHandler.DoProtocolSpecificBeginProcess(): void @060004A1

    Φ. Microsoft.Exchange.HttpProxy.AutodiscoverProxyRequestHandler OnlnitializingHandler(): void @060004A0

                                           ▶ @ Microsoft.Exchange.HttpProxy.AutodiscoverProxyRequestHandlerParseRequest(Stream): bool @060004A7
                                          ▶ ⊕ Microsoft.Exchange.HttpProxy.AutodiscoverProxyRequestHandler.ResolveAnchorMailbox(): AnchorMailbox @060004A3
                                          ▶ @ Microsoft.Exchange.HttpProxy.ComplianceServiceProxyRequestHandler.ComplianceServiceProxyRequestHandler(): void @060004D6
▶ @ Microsoft.Exchange.HttpProxy.EcpProxyRequestHandler.ResolveAnchorMailbox(): AnchorMailbox @0600050C
                                          ▶ Ø, Microsoft.Exchange.HttpProxy.EwsAutodiscoverProxyRequestHandler.EwsAutodiscoverProxyRequestHandler(): void @06000520
                                           ^{\bullet} \Phi_{\bullet} \  \, \text{Microsoft.Exchange.HttpProxy.EwsAutodiscoverProxyRequestHandler.ResolveAnchorMailbox(): AnchorMailbox(): AnchorMailbox():
                                          ▶ © Microsoft.Exchange.HttpProxy.EvsSProxyRequestHandler.BuildRequestStreamProxy(StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:StreamProxy:Strea
                                          ▶ ⊕ Microsoft.Exchange.HttpProxy.EwsProxyRequestHandler.OnlnitializingHandler(): void @0600052F
                                          Microsoft.Exchange.HttpProxy.EwsProxyRequestHandler.WillContentBeChangedDuringStreaming : bool @17000124
```

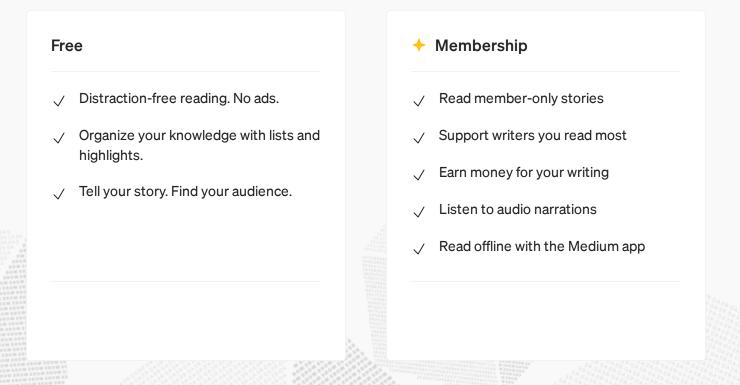
Microsoft.Exchange.HttpProxy.AutodiscoverProxyRequestHandler

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

And "/autodiscover" also allow for unauthenticated

```
private IHttpHandler SelectHandlerForUnauthenticatedRequest(HttpContext httpContext)
401
402
                  IHttpHandler result;
484
                     if (HttpProxySettings.NeedHandleAsAuthenticatedRequest(httpContext.Request.Headers, httpContext.Request.Cookies, h
495
407
                         result = this.SelectHandlerForAuthenticatedRequest(httpContext);
408
409
410
                         UriBuilder uriBuilder = new UriBuilder(httpContext.Request.Url);
411
412
                         string explicitLogonUser = null;
413
                         if (Microsoft.Exchange.Clients.Common.UrlUtilities.TryGetExplicitLogonUser(httpContext.Request, out explicitLo
414
415
                             uriBuilder.Path = Microsoft.Exchange.Clients.Common.UrlUtilities.GetPathWithExplictLogonHint(httpContext.R
416
417
                          IHttpHandler httpHandler = null;
418
                         if (HttpProxyGlobals.ProtocolType == ProtocolType.Autodiscover)
419
428
                             httpHandler = new AutodiscoverProxyRequestHandler();
421
                          else if (HttpProxyGlobals.ProtocolType == ProtocolType.Ews)
422
```

## **Medium**



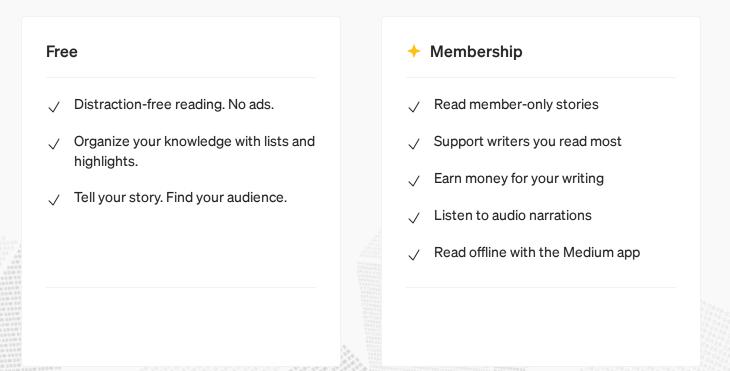
```
To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy,
        including cookie policy.
                            UrlAnchorMailbox urlAnchorMailbox = this.AnchoredRoutingTarget.AnchorMailbox as UrlAnchorMailbox;
                            if (urlAnchorMailbox != null)
                                result = urlAnchorMailbox.Url;
                                UriBuilder clientUrlForProxy - this.GetClientUrlForProxy();
                                clientUrlForProxy.Scheme = Uri.UriSchemeHttps;
clientUrlForProxy.Host = this.AnchoredRoutingTarget.BackEndServer.Fqdn;
clientUrlForProxy.Port = 444;
if (this.AnchoredRoutingTarget.BackEndServer.Version < Server.E15MinVersion)
2747
2749
                                     this.ProxyToDownLevel = true;
2752
2753
2754
2755
                                     RequestDetailsLoggerBase<RequestDetailsLogger>.SafeAppendGenericInfo(this.Logger, "ProxyToDownLevel", true); clientUrlForProxy.Port = 443;
                                result = clientUrlForProxy.Uri;
2759
2769
2761
2762
                       finally
                           this.LogElapsedTime("L_Target8EUrl");
                       return result:
```

ProxyRequest Handler. Get Target Back End Server Url ()

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 <u>here</u>

*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

# **Medium**



How can we get explicit ogen Address variable from our request?

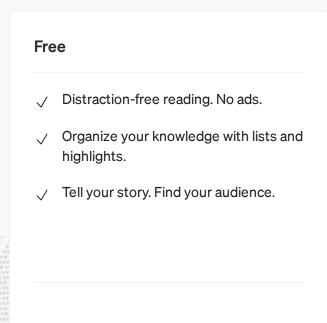
To make Medium work, we log user data. By using Medium, you agree to our <u>Privacy Policy</u>, including cookie policy.

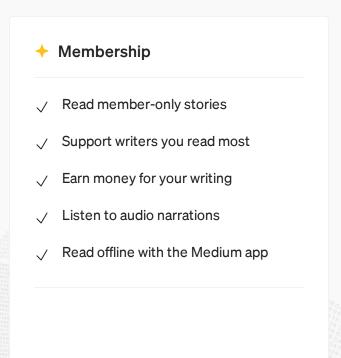
Notice that, Params variable contains parameters from query string, form parameter, cookies, ...

We need to pass some conditions

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

# Medium





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

To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy, including cookie policy.

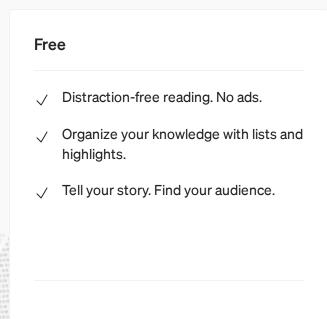
So it is "/autodiscover/autodiscover.json?a=dummy@dummy.pw" (in order to help us can reach the if statement which will return *False* and *remove explicitLogon*) and then we set this value into Email Cookie with the same value

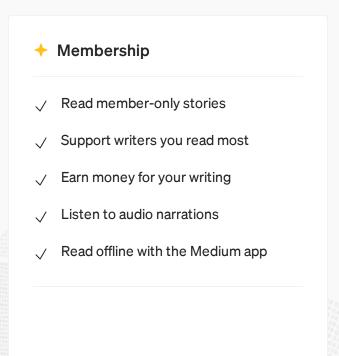
3. When preparing request to send to backend internal, Exchange will generate Kerberos auth header and attach into Authorization header. This is why we can reach some other endpoint without any authentication

PrepareServerRequest()

Chaining into together we have an pre-auth SSRF

# **Medium**





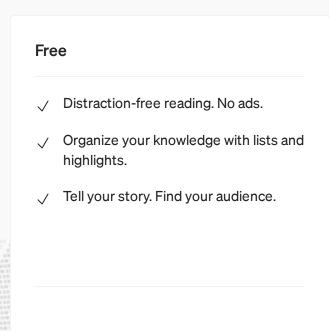
We don't have permission on this endpoint:(

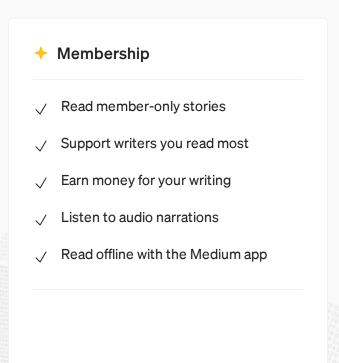
Always remember one think, you should understand on what you are looking while doing 1-day anlysis. IIS has some modules on each web application, they are excuted before the actual handler executed. You can imagine they're like "filter" mechanism on Java web apps.

**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.

#### **Medium**





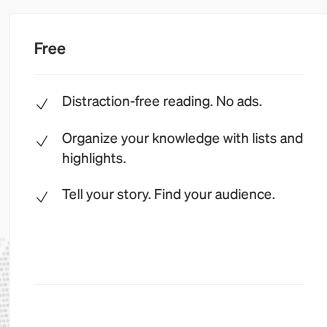
BackendRehydrationModule.ProcessRequest()

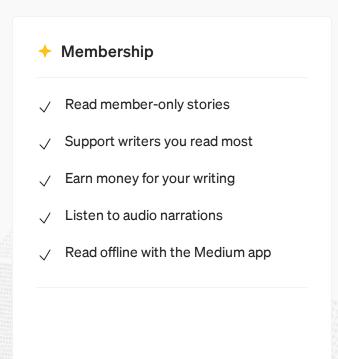
So how can we set the header "X-CommonAccessToken" because we cannot make Exchange copy it to SSRF request and send to "/powershell"

some blacklist cookies Exchange won't copy to internal

Before BackendRehydrationModule executed, there's RemotePowershellBackendCmdletProxyModule

# **Medium**





Basically, when the SSRE doesn't contain Header "Y. Common Access Tolzon"

Ex To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy, including cookie policy.

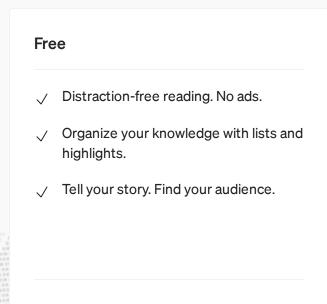
BackendRehydrationModule we will survive from the Exception. But how can we create a valid CommonAccessToken or maybe high privilege CommonAccessToken? We need to reverse the structure of CommonAccessToken

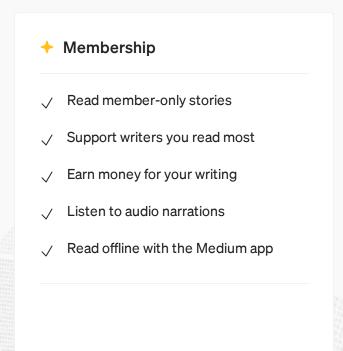
deserialize "X-Rps-CAT" into CommonAccessToken

```
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

## **Medium**





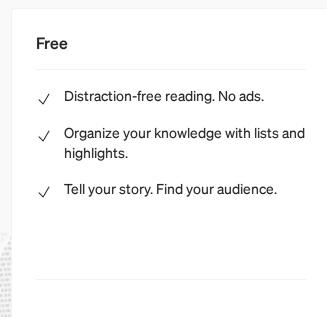
finally foward into 1/1/2 With this cotup we can canture a "cample'

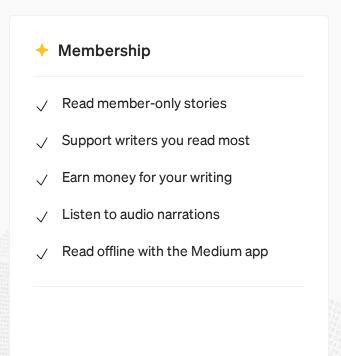
To make Medium work, we log user data. By using Medium, you agree to our <u>Privacy Policy</u>, including cookie policy.

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?

# Medium



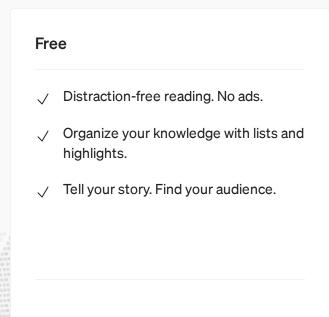


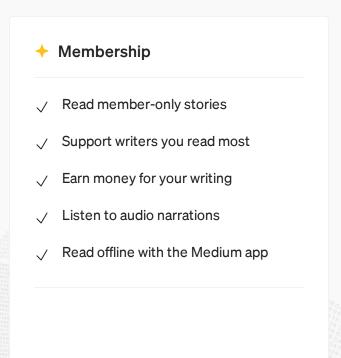
 $\frac{https://docs.microsoft.com/en-us/powershell/module/exchange/new-mailboxexportrequest?}{view=exchange-ps}$ 

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.

# Medium





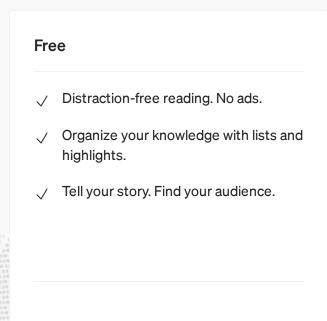
 $\frac{\text{https://docs.microsoft.com/en-us/openspecs/office_file_formats/ms-pst/5faf4800-645d-49d1-9457-}{2ac40eb467bd}$ 

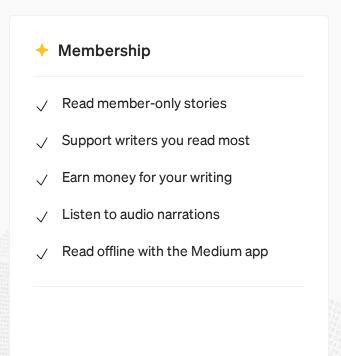
But how can we put our shell into the mailbox and then export it as our shell?

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 <u>EWS Impersonation</u>. Then injecting our "encoded" shell as an attachment.

# **Medium**





#### Channing all together

To make Medium work, we log user data. By using Medium, you agree to our <u>Privacy Policy</u>, including cookie policy.

"/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.

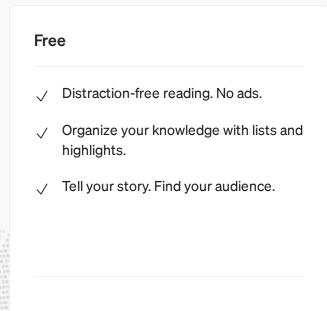
For myself, I use <u>pypsrp</u> then collect the data while it processing and plug it into our SSRF. To understand more about WinRM you can check this <u>awesome blog</u>

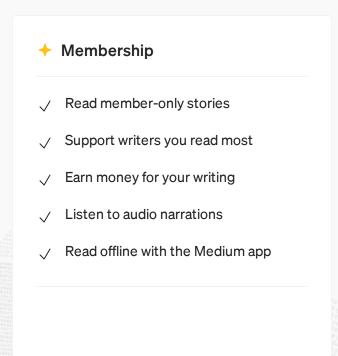
Or you can do the same with <u>Orange's way</u>, implement his own proxy to communicate with WinRM

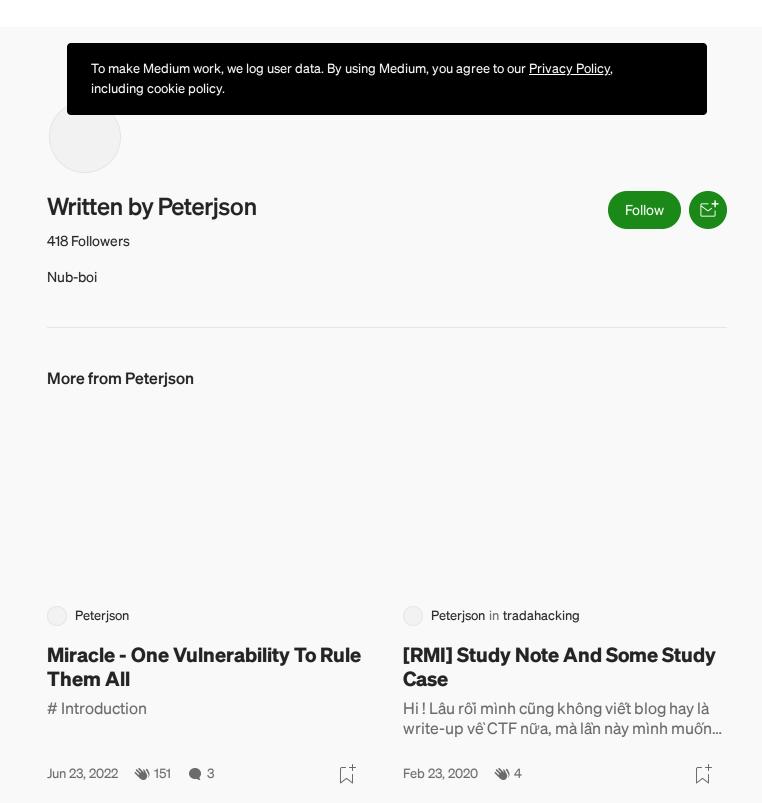
#### **Our demonstration:**

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

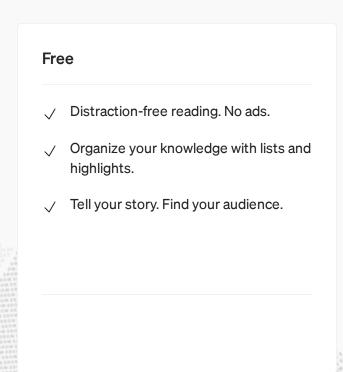
# **Medium**

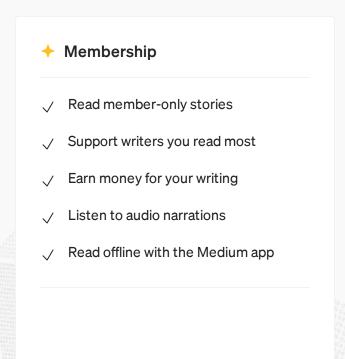




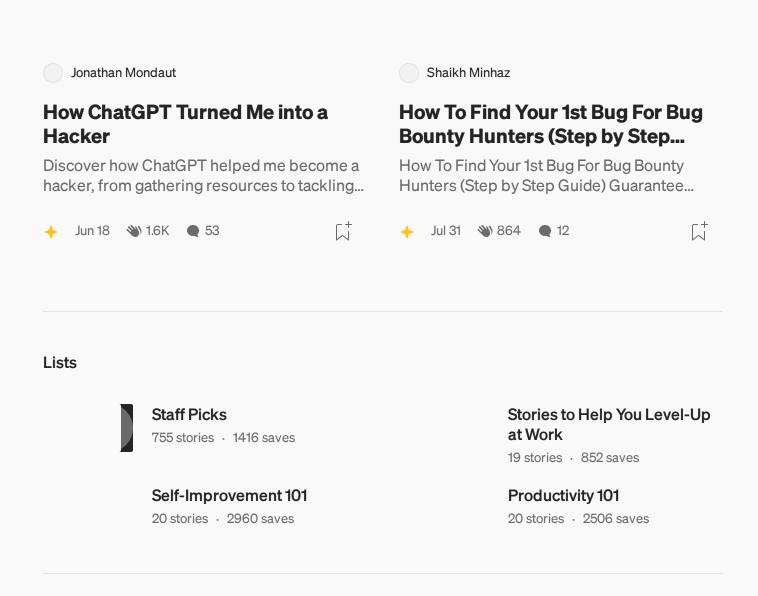


# **Medium**

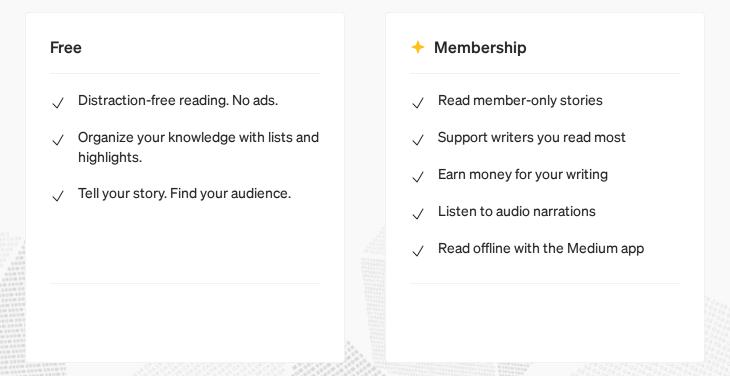




#### **Recommended from Medium**



# Medium



To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy,

including cookie policy. **RED TEAM** jayjonah.eth **Malware Development Part 8:** \$150,000 Evmos Vulnerability **Reverse Shell Via DII Hijacking Through Reading Documentation** "From DLL to Shell: A Step-by-Step Guide to Life as a Web3 security researcher often Reverse Shell via DLL Hijacking" consists of deep diving into technical subjec...  $\Box$ K Jun 22 🔌 147 5d ago **3** 64 **1** See more recommendations Help Status About Careers Press Blog Privacy Terms Text to speech Teams

# **Medium**

